

CLIPPEDIMAGE= JP406266596A

PAT-NO: JP406266596A

DOCUMENT-IDENTIFIER: JP 06266596 A

TITLE: FLASH MEMORY FILE STORAGE DEVICE AND INFORMATION
PROCESSOR

PUBN-DATE: September 22, 1994

INVENTOR-INFORMATION:

NAME

KATAYAMA, KUNIHIRO

KAKI, KENICHI

KITAHARA, JUN

HIDA, YASUHIRO

FURUSAWA, KAZUNORI

ASSIGNEE-INFORMATION:

NAME

COUNTRY

HITACHI LTD

N/A

APPL-NO: JP05051041

APPL-DATE: March 11, 1993

INT-CL (IPC): G06F012/00;G06F012/00

ABSTRACT:

PURPOSE: To attain plural byte parallel and simultaneous writing while efficiently using a storage area.

CONSTITUTION: A file storage device using a flash memory element whose minimum deleting unit is larger than X bits, and whose data access width is (x) bits ($x=X/p$; (p) is an integer ≥2) through an X bit data bus, is equipped with a flash memory device 5 constituted of the (p) sets of flash memory element groups to which an access are simultaneously possible, dividing means 51-54 which divides data on a data bus into at least (p) parts; data distributing

FH 008676

means 61 which has at least a first function which makes the (p) pieces of (x) bit data correspond to one set of the flash memory element groups, and a second function which makes the (p) pieces of (x) bit data respectively correspond to the different sets of the flash memory element groups, and control means 56 which controls the data distributing circuit 61 so that the first and second functions can be switched according to the number of the storage capacity units of the file management of the access object files.

COPYRIGHT: (C)1994,JPO&Japio

FH 008677

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平6-266596

(43) 公開日 平成6年(1994)9月22日

(51) Int. Cl. ⁵	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 12/00	5 2 0 P	8944-5B		
	5 5 0	9366-5B		

審査請求 未請求 請求項の数11 O L (全 21 頁)

(21) 出願番号 特願平5-51041

(22) 出願日 平成5年(1993)3月11日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 片山 国弘

神奈川県横浜市戸塚区吉田町292番地 株式会社日立製作所マイクロエレクトロニクス機器開発研究所内

(72) 発明者 柿 健一

神奈川県横浜市戸塚区吉田町292番地 株式会社日立製作所マイクロエレクトロニクス機器開発研究所内

(74) 代理人 弁理士 富田 和子

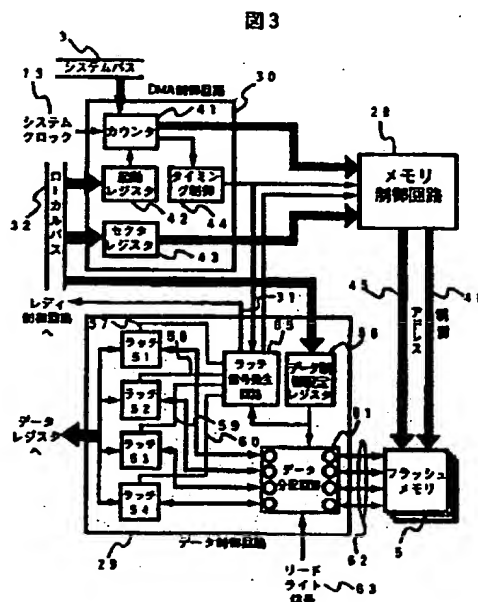
最終頁に続く

(54) 【発明の名称】 フラッシュメモリファイル記憶装置および情報処理装置

(57) 【要約】 (修正有)

【構成】 X ビットのデータバスを介し最小消去単位が X ビットより大きくデータアクセス幅が x ビット ($x = X/p$: p は2以上の整数) であるフラッシュメモリ素子を用いるファイル記憶装置において、それぞれ同時にアクセス可能な p 組のフラッシュメモリ素子群により構成したフラッシュメモリ装置5、データバス上のデータを少なくとも p 分割する分割手段51~54、少なくとも、 p 個の x ビットデータをフラッシュメモリ素子群の1組に対応付ける第1の機能、及び p 個の x ビットデータをそれぞれフラッシュメモリ素子群の別個の組に対応付ける第2の機能とを有するデータ分配手段61、並びにアクセス対象ファイルのファイル管理の記憶容量単位の数に応じて、第1及び第2の機能を切替るようデータ分配手段61を制御する制御手段56を備える。

【効果】 記憶エリアを有効に利用しながら複数バイト並列同時書込みが可能になる。



1

【特許請求の範囲】

【請求項1】ファイルのデータをXビットのデータバスを介して格納するファイル記憶装置であって、最小消去単位が前記Xビットより大きくデータアクセス幅がxビット ($x=X/p$: pは2以上の整数) であるフラッシュメモリ素子を用いるフラッシュメモリファイル記憶装置において、

それぞれ同時にアクセス可能なp組のフラッシュメモリ素子群により構成したフラッシュメモリ装置と、

前記Xビットのデータバス上のデータを少なくともp分割する分割手段と、

少なくとも、該分割により得られたp個のxビットデータを前記フラッシュメモリ素子群の1組に対応付ける第1の機能と、前記p個のxビットデータをそれぞれ前記フラッシュメモリ素子群の別個の組に対応付ける第2の機能とを有するデータ分配手段と、

アクセス対象ファイルのファイル管理の記憶容量単位の数に応じて、前記第1および第2の機能を切替るよう前記データ分配手段を制御する制御手段と、

を備えることを特徴とするフラッシュメモリファイル記憶装置。

【請求項2】前記制御手段は、前記フラッシュメモリ素子の最小消去単位のメモリエリア内には異なるファイルのデータが混在しないよう、前記データ分配手段を制御することを特徴とする請求項1記載のフラッシュメモリファイル記憶装置。

【請求項3】前記制御手段は、前記ファイルの連続するp個の記憶容量単位のデータに対して前記第2の機能を選択し、1個の記憶容量単位のデータに対して前記第1の機能を選択するよう、前記データ分配手段を制御することを特徴とする請求項1または2記載のフラッシュメモリファイル記憶装置。

【請求項4】前記記憶容量単位と、前記フラッシュメモリ素子の最小消去単位が等しいことを特徴とする請求項1または2記載のフラッシュメモリファイル記憶装置。

【請求項5】プログラムおよびデータを処理する中央演算処理手段と、該中央演算処理手段を駆動するクロック発振手段と、フラッシュメモリを記憶媒体としたファイル記憶手段と、該ファイル記憶手段のフラッシュメモリのアクセスを制御するファイル記憶制御手段を構成要素とする情報処理装置において、

前記クロック発振手段が発生するクロック信号と同一の信号あるいは同期する信号を前記ファイル記憶制御手段に入力し、前記中央演算処理手段と該ファイル記憶制御手段が同期動作を行ってファイルデータを送受することを特徴としたフラッシュメモリファイル記憶装置を搭載する情報処理装置。

【請求項6】前記中央演算処理手段と前記ファイル記憶手段の1回のデータアクセス幅が異なる場合、前記ファイル記憶制御手段はファイルデータの処理中か処理完了

2

かを示す状態提示信号を出力する手段を備え、前記中央演算処理手段は前記状態提示信号を受け、これが処理中であることを示していれば処理終了までの期間、処理の進行を停止する機能を有し、前記ファイル記憶制御手段は前記ファイル記憶手段が扱うデータビット数を前記中央演算処理手段の処理データビット幅に合わせるデータビット幅制御手段を有し、該データビット幅制御手段はデータビット幅を一致させるために必要とする期間、前記状態提示信号により処理中であることを前記中央演算処理手段に示して待機させ、双方の処理データ幅を一致させてデータのやり取りを行うことを特徴とした請求項5記載の情報処理装置。

【請求項7】前記中央演算処理手段のデータアクセス幅に等しいビット数のデータを生成するために必要な数のフラッシュメモリ群により前記ファイル記憶手段を構成し、それぞれのメモリ群にシーケンシャルな順序を規定し、複数の記憶容量単位数に及ぶファイルの格納においてはこの規定されたメモリ群の順序に従うこととし、一旦格納されたファイルを更新することにより記憶容量単位数が増加する場合には該増加に対して、前回格納したときにファイルの最後のデータが格納されていたメモリ群の次の順序に規定されているメモリ群から増加記憶容量単位分の記憶エリアを確保することを特徴とする請求項6記載の情報処理装置。

【請求項8】中央処理装置と、ファイル記憶装置を備える情報処理装置において、

前記ファイル記憶装置の記憶媒体として、複数のフラッシュメモリ素子を用い、該フラッシュメモリのデータの最小消去単位をファイル管理の記憶容量単位と等しくし、かつ、

前記中央処理装置がファイルアクセスを要求する際の記憶容量単位数が複数である場合は前記複数のフラッシュメモリ素子を同時にアクセスし、記憶容量単位数が1である場合には、前記複数のフラッシュメモリ素子の1個をアクセスすることを特徴とした情報処理装置。

【請求項9】フラッシュメモリを記憶媒体とするファイル記憶装置において、記憶するある1つのファイルの記憶容量が複数の記憶容量単位に及ぶ場合には、各記憶容量単位ごとに、次に続く記憶容量単位が格納されている物理的な位置の情報を記憶する連鎖情報記憶手段を備えたことを特徴とするフラッシュメモリファイル記憶装置。

【請求項10】1つのパッケージに複数のメモリチップが組み込まれているメモリ素子において、

前記メモリパッケージが全メモリチップのデータ入出力数の総計に相当する数の入出力データ端子を備え、また各メモリチップの入出力データとデータの入出力端子を切り換え接続制御するデータ制御手段と、該データ制御手段に対するメモリ使用者からの指示のための制御信号端子を備え、該制御信号端子の指示により前記複数のメ

3

メモリチップのデータを切り換え接続して前記入出力データ端子の任意の端子に任意のメモリチップの入出力データを接続することを可能としたメモリ素子。

【請求項11】 1つのパッケージに複数のメモリチップが組み込まれているメモリ素子において、前記メモリパッケージが全メモリチップのデータ入出力数の総計に相当する数の入出力データ端子を備え、また各メモリチップの入出力データとデータの入出力端子を切り換え接続制御するデータ制御手段と、該データ制御手段に対するメモリ使用者からの指示のための制御コマンド設定手段を備え、該制御コマンド設定手段の指示により前記複数のメモリチップのデータを切り換え接続して前記入出力データ端子の任意の端子に任意のメモリチップの入出力データを接続することを可能としたメモリ素子。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、フラッシュメモリを記憶媒体とするファイル記憶装置を搭載した情報機器に係り、特に高速のファイルアクセスを要求する情報機器に適したファイル記憶装置のファイルアクセス方式に関する。

【0002】

【従来の技術】 パーソナルコンピュータを始めとする今日の汎用の情報機器では補助記憶装置はほとんど必須といえる周辺機器である。しかも、補助記憶装置は、本体の機器に内蔵されているのが一般的であり、ユーザは常に大容量のファイルを扱うことができるようになっている。

【0003】 最近では、ノートサイズやバームトップ型のパーソナルコンピュータが使われることが多くなってきており、移動して使用する用途が重要視され始めている。このため振動に弱く消費電力が大きい磁気ディスク記憶装置のかわりに半導体を記憶媒体としたファイル記憶装置が注目を浴びている。例えば、特開平2-292798号公報には、記憶媒体としてフラッシュメモリを使用した半導体ファイル記憶装置の技術が開示されている。

【0004】 フラッシュメモリとは、電気的書換え可能な不揮発性メモリであり、大容量低価格化が可能であるため、半導体ファイル記憶装置の記憶媒体として最も有効なメモリの一つである。先記の特開平公報技術は、このメモリを使用してファイル記憶装置を構築する際の問題となる点を解決し、使い勝手を向上させる工夫をしている。例えば書換の頻度により素子が劣化するというフラッシュメモリの欠点を救済する方法や、フラッシュメモリの書換えに必要な消去というフェーズにおける高速化の手法などを提案している。また、ホストとなる情報機器とのインタフェースとしては磁気ディスク装置と同一であることを提案しており、磁気ディスク装置に代

4

替できるシステムの構築を目的としている。

【0005】

【発明が解決しようとする課題】 上記従来技術の半導体ファイル記憶装置はホストとなる情報機器の既存のインタフェースバスを使用し、磁気ディスク装置とのコンパチビリティを重視している。これにより半導体ファイル記憶装置をユーザに抵抗なく受け入れられるようにしているが、磁気ディスク装置との互換性を重視する反面、半導体記憶素子の磁気ディスク装置に対する優位性の利用に考慮がなされていない。

【0006】 すなわち、半導体記憶素子は、回転するディスクからデータを拾い上げ書き込みを行う磁気ディスク装置と異なり、静的な記録媒体なので非常に高速なデータアクセスを可能とするが、磁気記憶装置と同一のインタフェースではこの高速アクセスの優位性を活かすことができないという問題がある。

【0007】 本発明は、半導体記憶素子を記憶媒体とすることにより、アクセスの高速性能を最大限に引き出すインタフェースを構築し、磁気ディスク装置の欠点といえるアクセス性能の向上を新しい補助記憶装置を提供することにある。

【0008】

【課題を解決するための手段】 本発明によるフラッシュメモリファイル記憶装置は、ファイルのデータをXビットのデータバスを介して格納するファイル記憶装置であって、最小消去単位が前記Xビットより大きくデータアクセス幅がxビット ($x = X/p$; pは2以上の整数) であるフラッシュメモリ素子を用いるフラッシュメモリファイル記憶装置において、それぞれ同時にアクセス可能なp組のフラッシュメモリ素子群により構成したフラッシュメモリ装置と、前記Xビットのデータバス上のデータを少なくともp分割する分割手段と、少なくとも、該分割により得られたp個のxビットデータを前記フラッシュメモリ素子群の1組に対応付ける第1の機能と、前記p個のxビットデータをそれぞれ前記フラッシュメモリ素子群の別個の組に対応付ける第2の機能とを有するデータ分配手段と、アクセス対象ファイルのファイル管理の記憶容量単位の数に応じて、前記第1および第2の機能を切替るよう前記データ分配手段を制御する制御手段とを備えることを特徴とする。

【0009】

【作用】 現状のパーソナルコンピュータに使用されている磁気ディスク装置では、データアクセスがパーソナルコンピュータのメモリアクセスに比較して遅く、パーソナルコンピュータ内のCPUと同期して動作する必要はない。そのため磁気ディスク装置のデータの授受は非同期的なバス上で行われている。半導体を記憶媒体とした場合にはCPUの動作に追従することができるため、同期動作にすることが有意義となる。

【0010】 しかしながらこの時の問題点として、CP

5

Uの処理のバス幅とフラッシュメモリ1チップのデータアクセスのバス幅が異なるという点がある。DRAMなどではこのバス幅の違いをメモリチップを並列して使用することにより解決できるが、フラッシュメモリでは消去単位が決まっており、最小の消去単位は例えば512バイトである。この場合、複数チップを並列に利用すると、一度に消去を行う消去単位の容量が(512×並列チップ数)バイトになってしまう。これでは、消去対象容量が大きすぎて、消去対象以外のデータまで消去してしまうことになる。

【0011】一方、現在、パーソナルコンピュータのファイル管理はこの512バイト単位を一区切り(1セクタ)としているため、このファイル管理の記憶容量単位(以下、ファイル管理単位ともいう)を変更することは避けるべきである。従って、ファイル管理単位である1セクタのアクセスにおいて例えばフラッシュメモリ4チップを並列に利用しようとする、結果的に最小消去単位が大きくなり、このセクタの書き時に前述のように1セクタの4倍の容量を同時に消去することが必要になる。このような事態を避けるために、例えばCPUが32ビットバス、フラッシュメモリが8ビットバスであれば、1チップを4回順次アクセスしてCPUの32ビットバスに対応する必要がある。

【0012】しかし、4セクタ以上の連続アクセス時には、各4セクタのデータについて、4チップに並列に同時アクセスして、1回のアクセスでCPUのデータバスからの32ビットを同時アクセスするようにしても問題ない。すなわち、最初に第1セクタの512バイトが32ビットずつ16回に分けて書き込むと、各チップに128(16×8)バイトが分散する。次に第2セクタの512バイト32ビットずつ16回に分けて書き込む。続いて、第3セクタの512バイトを、さらに第4セクタの512バイトを書き込む。この時点で、4チップの最小消去単位の容量がすべてデータで満たされることになる。したがって、このファイルの書き時にはこの4セクタ分の容量を一括して消去して問題ない。

【0013】このようにして、ファイルを4セクタずつ同時4チップ並列アクセスを繰返す。最後の4セクタに満たないデータが残った場合には、別の対処が必要となる。

【0014】3セクタ残った場合には、CPUの32ビットが3で割り切れないので、2セクタと1セクタに分けて対処する。

【0015】2セクタ残った場合、あるいは、初めから2セクタのファイルである場合、CPUからの32ビットのデータを2分割し、一度に16ビットずつ2チップに同時アクセスする。第1セクタの512バイトは、16ビットずつ32回に分けて2チップに書き込まれ、各チップに256(32×8)バイトずつに分散することになる。次に、第2セクタの512バイトが同様に16

6

ビットずつ32回に分けて2チップに書き込まれる。

【0016】1セクタ残った場合、あるいは初めから1セクタのファイルの場合、CPUからの32ビットのデータを4分割し、一度に8ビットずつ1チップにアクセスする。1セクタの512バイトは64回に分けてどのチップの1セクタエリアに格納される。

【0017】以上のように、連続アクセスするセクタ数によりデータの格納方法を変え、アクセスの高速化に寄与するとともにフラッシュメモリに伴う消去の問題を解決できる。すなわち、いずれの格納方法においても、1チップの最小消去エリア内には同一のファイルのデータのみが格納されることが保証されるので、フラッシュメモリの消去に伴う上述の問題が解消される。

【0018】なお、本発明では、フラッシュメモリを同期動作させるが、アクセス形式によって遅延が生じる。この問題に対しては、CPUにレディ信号入力があるものを用いることにより、任意にCPUに待機状態を要求できる。レディ信号入力のあるCPUは今日では一般的である。例えば現在の汎用のパーソナルコンピュータに搭載されているCPUとして最も一般的なINTEL社の16ビット処理以上のCPUには全て備えられている。このレディ信号をネゲートすることによりCPUは処理サイクルを停止するため、アクセスデータの読み出し、書き込みが終了していなければこの信号をネゲートするだけで良く、アクセスが終了したらアサートすればCPUは処理の実行を再開する。これらはファイル記憶装置がCPUと同期動作をして初めて可能となる制御であるため、この時に同期したクロックを双方に入力して同期動作させることは必須である。これらの制御により一回の処理におけるデータ幅がCPUとフラッシュメモリで異なる場合にも、データ幅が揃うまでCPUを待機させることにより容易に対応可能である。

【0019】ファイル管理上一度連続セクタで扱ったものを後から分割することは通常あり得ない。すなわち、ファイルシステムはファイル単位でしかアクセスされない。したがって、複数セクタの連続アクセスの場合は書き込みの時点で複数チップを同時アクセスして複数セクタを並列使用するようにすれば、読み出しの際に同様の並列アクセスをすれば書き込んだままのファイルのデータを得ることができる。

【0020】以上のように、本発明によればフラッシュメモリを記憶媒体とするファイル記憶装置を搭載した情報処理装置において、磁気ディスク装置と比較して優位性が高いアクセス性能において、最大限の性能を引き出せるようシステムとの同期制御を行う。その際にシステムのCPUに入力するレディ信号により適宜CPUを待機状態においてタイミングを合わせる。システムのデータバス幅とフラッシュメモリ1チップのアクセスデータビット数が異なる場合でも、できるかぎり高速にアクセスすることが可能になる。

7

【0021】特にアクセス速度がユーザに認識されやすい大容量のファイルのリードライトすなわち連続セクタアクセスにおいて効果大きい。また、システムのデータバス幅とフラッシュメモリチップのアクセスデータビット数は用途、性能、時代によって異なるが、様々なものに対応できる柔軟性の高い構成を可能とする。またメモリをインタリープアクセスして高速化する方法にも応用できる。

【0022】連鎖情報を備える構成によれば、システムからのアクセス手順を簡略化し高速アクセスを援助する。そしてファイル管理自体を簡略化でき、コントロール回路や制御プログラムなどの簡素化が図れる。

【0023】また、メモリ素子の中にデータ分配機能を備えることにより、周辺回路の削減、データ処理の高速化が図れる。

【0024】

【実施例】以下、本発明における実施例を図を用いて詳細に説明する。

【0025】まず、図1に標準的なパーソナルコンピュータの構成に本発明のフラッシュメモリファイル装置を20 設置したものを示す。図中、1はデータやプログラムの処理を司るCPUであり、そのデータバス幅は32ビットであるものとする。2はシステム全体の同期クロックを発生するクロック発生器である。3はシステム内部のバスでデータバス、アドレスバス、メモリコマンド、I/Oコマンドなどを含む。4は本発明のフラッシュメモリファイル装置のファイル管理やメモリ制御を行うファイル制御回路である。5は、フラッシュメモリファイル装置の記憶媒体となるフラッシュメモリアレイであり、1チップのアクセスデータビット数は8ビットであるとする。6はシステムの主記憶を管理制御する主記憶制御回路、7は主記憶であるDRAMである。8は周辺IOバスの制御回路であり、周辺IO装置の一つとして表示制20 御回路9と表示装置10が接続されている。この他の周辺IO装置として、通信装置11や外部大容量記憶装置12などが挙げられる。

【0026】通常、周辺IOバス制御回路8には別のクロック発生回路が内蔵されており、これら周辺IO装置はそのクロックの周期に基づいて動作する。しかし高速化のために内部システムバス3に直接接続して、CPU1と同期動作するものも考えられる。13は、内部システムバス3に接続された回路を同期させるためにCPU1を含む各回路に供給されるクロック信号である。但し、CPU1に与えられるクロックと全く同一である必要はなく、回路によっては分周したもので同期していれば良い。

【0027】14は、CPU1に入力されるレディ制御信号であり、各回路より出力された状態提示信号をレディ制御回路15で統括してCPU1に入力される。16は、本システムの使用者が所望の処理を指示するための

8

入力装置の制御回路であり、17は入力装置である。図では入力装置17をキーボードとし、その制御回路16はキーボードコントローラ(KBDC)としている。

【0028】次に、図1のシステムの動作を説明する。通常の動作時は、使用者から入力装置17により指示された処理を実行すべく主記憶7に格納されているプログラムやデータをCPU1が演算処理し、その結果を表示装置10に表示する。また必要であれば通信装置11を起動したり、大容量外部記憶装置12へ大容量データの格納を行う。そして、ファイルを引き出したり格納したりという動作の際にはファイル制御回路4とフラッシュメモリ5からなるフラッシュメモリファイル装置を動作させる。システムの立ち上げ時もここからシステムプログラムをロードすることになる。これらの動作の際には、クロック発生器2が発生するクロック13により同期動作し、ある回路がCPU1に待機を要求する必要が生じたら、その回路がレディ制御回路15にCPU待機を要求し、レディ信号14のネゲートによりCPU1にそれが伝えられる。CPU1はレディ信号14が再度アサートされるまで待機を続けることになる。この時、ファイル制御回路4はCPU1に要求されるファイル数によりCPU待機時間を増減する制御を行えるようになっている。この仕組みを図2により説明する。

【0029】図2は、フラッシュメモリファイル装置の内部構成を説明する図である。図中、3、4、5、13は図1と同様のものであり、以下はファイル制御回路4内部の構成要素である。21は、システムバス3とインタフェースを行うレジスタ群であり、22はファイル制御回路4の状態をCPU1に報告するためのステータスレジスタ、23はアクセスするスタートセクタ番号の設定レジスタ、24はアクセスする最後のセクタ番号の設定レジスタ、25はCPU1が要求する処理をコマンドコードとして指示するコマンドレジスタ、26はシステムバスとのデータの授受を行う窓となるデータレジスタである。27は、ファイル制御回路4内部の制御を統括するコントローラであり、ワンチップマイコンなどのプログラム可能なインテリジェントなLSIが理想的である。28は記憶媒体であるフラッシュメモリ5の制御を行う制御回路、29はフラッシュメモリに書き込みあるいは読み出すデータの制御を行うデータ制御回路である。30はメモリのアクセスを高速に行うためのDMA制御回路であり、システムのクロック信号13は本回路に入力される。31はレディ制御回路15に出力する状態提示信号、32はこのフラッシュメモリファイル装置内のローカルバス、33はフラッシュメモリをアクセスするための制御信号及びアドレスである。

【0030】次に図2のフラッシュメモリファイル装置の動作を説明する。CPU1はフラッシュメモリファイル装置へのアクセスの必要性が生じたら、システムバス3を通してアクセスを行う。それにはまずステータスレ

ジスタ22の内容を読み出してアクセス可能な状況を確認する。そして次にアクセスするセクタをスタートセクタレジスタ23とエンドセクタレジスタ24に設定する。そして要求するアクセスのコマンドコード（リードまたはライト）をコマンドレジスタ25に書き込む。そして再びステータスレジスタ22を読み出し、アクセス可能であればデータレジスタ26へのデータの書き込みあるいは読み出しを行う。

【0031】この際、コントローラ27はこれらインタフェースレジスタ21の管理を行い、CPU1からの要求に応えるようにする。つまりスタートセクタレジスタ23やエンドセクタレジスタ24、コマンドレジスタ25を読み取ってフラッシュメモリ5のアクセス内容を把握し、現在の状態を示すコードをステータスレジスタ22に書き込んでCPU1に報告する。

【0032】コントローラ27はCPU1からのアクセス要求に応えるためのフラッシュメモリアクセスを直接行うには動作速度が遅いことが考えられるので、DMA制御回路30により高速にフラッシュメモリアクセスを行い、システムバス3とのデータの授受を行う構成としている。そのためのDMA制御回路30やメモリ制御回路28へのアクセス内容の設定やDMAの起動などをコントローラ27が行うことになる。

【0033】DMA制御回路30はDMAを行うためのアドレス発生及びタイミング生成を行い、メモリ制御回路28はそのタイミングに従いアクセス信号を発生する。フラッシュメモリ5はこれらの入力信号によりデータ制御回路29とデータの授受を行う。データ制御回路29はアクセスのセクタ数に応じてデータの生成を行う。

【0034】例えば、1セクタのライトアクセスであればシステムバス3からの1回のアクセスにより送られるデータをフラッシュメモリ5の1チップの書き込みビット数に合致させる。本実施例では、システムバス3のデータバス幅を32ビットとしているため1回のアクセスによる転送で32ビットのデータが得られる。そしてフラッシュメモリ1チップのデータ幅を8ビットとしているので送られたデータを4回に分けてフラッシュメモリ5に書き込むことになる。そのために32ビットデータを4つの8ビットデータへ分割する処理を、データ制御回路29がラッチ回路を用いて行う。また逆に1セクタのリードアクセスであれば、フラッシュメモリ5を4回リードアクセスして32ビットデータを用意して、1回のバス転送とする。この際に生ずるシステムバス3の待ち時間はデータ制御回路29が発生する状態提示信号31によるCPU1への待機要求で生じさせる。

【0035】一方、複数セクタのアクセスの場合は、データ制御回路29がラッチするデータを調節してアクセスを高速化して正常実行する。例えば4セクタの連続リードアクセスであれば、8ビットアクセスのフラッシュ

メモリ5を4チップ同時にリードすることにより32ビットのデータが完成するため、システムバス3の待ち時間を大きく減らすことができる。ただしフラッシュメモリ5への書き込みの時点で4チップ同時ライトアクセスを行っておく必要がある。そうしないとデータの順番が異なってしまう正常なファイルデータではなくなる。ただし、ファイル記憶装置を扱うシステムでは、ファイル単位で管理を行うのが一般的であるため、書き込んだときのセクタ数と同じセクタ数をリードアクセスするのが普通であり、セクタ数によるアクセスの形式をライトとリードで全く等しくしていれば先述の情報を記録する必要はない。つまり、例えば5セクタの連続ライトの際には最初の4セクタは4チップに並列に同時書き込みし、残る1セクタは1チップに4分割する、というアクセス形式をリードとライトのいずれにおいても採用すれば常に正常なファイルデータをアクセスできる。なお、念のために、フラッシュメモリ5に格納されているセクタ単位のデータに、データ格納形式の情報を記録しておいてもよい。その記録場所としてはフラッシュメモリのデータ格納領域以外の冗長領域が存在すればそこに格納するのが適当であり、冗長領域がない場合は別の記憶領域を設置して記録する。

【0036】6セクタの連続アクセスに際しては、4セクタの並列アクセスと2セクタの並列アクセスにより処理を行う。

【0037】次に、これらのアクセス信号およびデータ制御方法を図3によりさらに詳しく説明する。

【0038】図3では、システムバスを32ビット、フラッシュメモリを8ビットのビット幅としたときの構成を示している。図中、既出の番号はこれまでに説明したものと同一のものである。新たに、41は、DMA制御回路30のアドレス発生のためのカウンタであり、システムクロック13やシステムバス3のIOアクセス（コマンド）信号あるいはメモリアクセス（コマンド）信号を入力し、これに同期してカウントアップする。42は、コントローラ27のローカルバス32が接続されるDMA制御の起動レジスタであり、このレジスタにコードを書き込むことにより所望のDMA転送を開始することができる。43は、やはりローカルバス32に接続されるセクタ番号レジスタであり、アクセスするセクタ番号を書き込むことにより任意のセクタ番号のDMA転送を行うことができる。実動作上は、このセクタ番号の書き込み値をメモリ制御回路28に入力してフラッシュメモリ5の上位アドレスやチップセレクト信号の生成に使用される。44は、DMA転送時に各制御回路で同期をとるためのタイミング信号を発生するタイミング回路である。45は、カウンタ41とセクタレジスタ43の値に基づいてメモリ制御回路28が発生するメモリアドレスである。46は、メモリアドレス45の発生に合わせて生成するメモリ制御信号である。51、52、53、5

11

4は、各々1バイト(8ビット)のデータラッチであり、データ制御回路29内における32ビットデータと8ビットデータとの間のデータ幅変換のための4バイト(32ビット)のデータラッチを構成している。システムバス3のデータをD0~D31とすると、ラッチ51はD0~D7、ラッチ52はD8~D15、ラッチ53はD16~D23、ラッチ54はD24~D31をそれぞれラッチする。55は、これらデータラッチのラッチ信号発生回路である。56は、ローカルバス32と接続されてデータ幅やデータの並び方を設定するデータ制御設定レジスタである。本実施例では、このデータ制御設定レジスタ56にリードアクセス時の連続アクセスセクタ数として“1”、“2”、“4”のいずれかを設定することによりラッチ信号の発生とそのタイミングをラッチ信号発生回路55に指示する。そしてデータラッチ51、52、53、54のそれぞれに入力するラッチ信号が57、58、59、60である。例えば“1”を設定すれば、ラッチ信号57、58、59、60を順次一つずつ出力し、1チップのフラッシュメモリからのデータを4回アクセスして32ビットのデータを生成してシステムバス3に出力する。また“2”を設定すれば、ラッチ信号57、58を同時に出力し、その後ラッチ信号59、60に同時に出力するということを交互に行い、2チップのフラッシュメモリからのデータを2回ずつアクセスして32ビットのデータを生成してシステムバス3に出力する。また“4”を設定すればラッチ信号57、58、59、60に同時に出力して、4チップのフラッシュメモリからのデータを1回だけアクセスして32ビットのデータを生成してシステムバス3に出力する。

【0039】一方、ライトアクセスの時は常にデータラッチ51、52、53、54に同時にラッチ信号を送ってシステムからの32ビットデータを一度にラッチすることになる。61は、データラッチ51、52、53、54に格納されたデータあるいはメモリ5からのデータを振り分けるデータ分配回路である。62は、フラッシュメモリ5とデータ分配回路61を結ぶ32ビットのデータバスである。63は、データ分配回路61のデータの方角を決定するためのリードライト信号であり、インタフェースレジスタ21のコマンドレジスタ25などから供給を受けるのが良い。データ分配回路61は双方向のパッファになっており、片側はデータラッチ51、52、53、54に接続され、もう一方はフラッシュメモリ5に接続されている。フラッシュメモリ5側では32ビット分の入力となっており、フラッシュメモリ5の全チップを4組に分けて1組ずつ別のビット群(ビット0~7)(ビット8~15)(ビット16~23)(ビット24~31)に分け、計32ビットとして入力している。そしてリードかライトかによって方向を定め、それぞれのデータの振り分けをデータ制御設定レジスタ56の設定内容によって決定する。

12

【0040】このデータの分配について図12、図13、図14により具体的に説明する。これらの図は、データ分配回路61のデータ分配例をデータ制御設定レジスタ56の設定値別に示したものである。それぞれ、図12は連続アクセスセクタ数が1、図13は連続アクセスセクタ数が2、図14は連続アクセスセクタ数が4、である場合におけるデータ分配を示しており、各場合に4つのメモリ群のどれにアクセスするかにより4種類のデータ分配がある。またリードライトも区別されている。連続アクセスセクタ数が1の場合には4つのシステムサイクルで1アクセスとなり、連続アクセスセクタ数が2の場合には2つのシステムサイクルで1アクセス、連続アクセスセクタ数が4の場合には1つのシステムサイクルで1アクセスとなる。

【0041】連続アクセスセクタ数が1の場合は4つに分割したメモリのどこにアクセスするかにより4種類に分けられるが、連続アクセス数が2及び4においては、どのメモリ群を起点にするかによりやはり4種類に分かれる。すなわちメモリの使われ方により起点を適宜決めることができ、このようにすることにより使用されるメモリ群が偏ることを防ぐことができる。例えば必ずメモリ群1を起点にするとデータの分配法は簡素化できるが、メモリ群1が使用される比率が高くなり使用量、使用頻度ともに偏りが生じることが予想できる。偏りが生じると最終的には、一つのメモリ群が使用できなくなり、連続セクタ数が4の高速書き込みアクセスができなくなってしまうことになる。従って起点は全てのメモリ群に設定可能とする。

【0042】なお、連続セクタ数が1または2のリードにおいては、図では分配回路のデータ結線をそれぞれのサイクルにおいて別々にしているが、ラッチ信号を目的のラッチ以外には出力しないため、各サイクルで分けることなく結線を一緒にしてもよい。つまり連続セクタ数が1(図12)のAのリードにおいては、分配を順次のサイクルで、ラッチ1-メモリ群1、ラッチ2-メモリ群1、ラッチ3-メモリ群1、ラッチ4-メモリ群1としているが、全てのサイクルで全てのラッチへの結線をメモリ群の1に接続しても構わない。これはデータの結線をしていてもラッチ信号を出力しなければ影響がないからである。

【0043】さらに具体的な説明を加えれば、例えば1セクタのライトアクセスでは図12を参照し、32ビットのデータバス62のうちアクセス対象のフラッシュメモリ群の1つに接続された1バイトを、1回目ではデータラッチ51に、2回目ではデータラッチ52に、3回目ではデータラッチ53に、4回目ではデータラッチ54に、以下51から順番に繰り返してフラッシュメモリへの書き込みを行う。リードにおいては、ラッチ信号により1サイクルずつ各ラッチにデータを振り分ける。

【0044】2セクタのライトアクセスでは、すなわち

13

データ幅2バイトのライトであれば、図13より32ビットのデータバス62のうち該当する2つのメモリ群に接続されている2バイトのデータとして1回目のアクセスではデータラッチ51、52より、2回目のアクセスでは53、54よりデータを受け取り、以下交互に振り分ける。一方、リードアクセスでは方向が逆になり、該当する2つのメモリ群から1回目のアクセスではデータラッチ51、52に、2回目のアクセスでは53、54にデータを振り分け、これを繰り返す。

【0045】4セクタのライトアクセス、すなわちデータ幅4バイトのライトであればバッファの方向はデータラッチからフラッシュメモリ5の方向となり、32ビットのデータバス62は先頭に当たるメモリ群から順にデータラッチ51、52、53、54に接続され、1サイクルのアクセスで32ビットのアクセスが完了する。リードでは方向が逆になる。

【0046】以上の動作を行うためにコントローラ27はDMA制御回路30に起動をかける前に、これまでに説明したレジスタ類に適切な値を設定しておく必要がある。

【0047】高速化のために、前述のように、複数のメモリ群に同時にアクセスを行う場合、各メモリ群に与えるアドレスについて、以下、図5を参照して検討する。

【0048】図5において、81~84は、それぞれメモリ群1~4を示している。同図(1)は、ある程度のファイルの書き込みが行われた状態を示し、同図(2)は、その後、あるファイルについて更新されそのファイル容量が増えた状態を示している。図示のデータ配号(m-n)のmはファイル番号、nは各ファイル内のセクタ番号である。例えば、(3-2)は、ファイル番号3番の2セクタめを示す。但し、この図は、各ファイルの1以上のセクタに対して、どのメモリ群のエリア(1セクタ分の容量を有する)がどのように利用されるかを総体的に示したものであり、実際には、2以上のセクタの連続アクセス時には、各メモリ群の1セクタ容量エリア内に単独のセクタ内容のみが格納されるのではなく、複数のセクタの内容が分散して格納される。この点に関しては、ファイル管理の説明において後に詳述する。

【0049】図5(1)において、ファイル番号1(1-nと付されている全てのセクタ)のアクセスにおいて、1-1から1-4までの同時アクセスにおいては、これらが同じ列に並んでいるため、各メモリ群に同一のアドレスを与えることができる。しかし、ファイル番号2においては、2-1から2-4までを同時アクセス使用すると、メモリ群4には他のメモリ群とは異なるアドレスを与える必要がある。これは、アドレスバスをメモリの数だけ設ける必要があることを意味する。あるいは、メモリ群に異なるアドレスを与えるなんらかの方法が必要となる。従って、これを避けるためには、ファイル番号2のアクセスにおいては、2-1は単独でアクセ

14

スし、2-2、2-3を同時アクセスし、さらに2-4は再度単独でアクセスするというアクセス形式をとる必要がある。しかし、これでは高速化においては無駄が生じる。そこで、異なるアドレスを与える構成例を図9により説明する。

【0050】図9は、メモリ群数を4としたときのメモリ制御回路28内のアドレス生成回路の構成例である。

図中、201~204はフラッシュメモリ5を構成する4つのメモリ群であり、201は第1のメモリ群a、202は第2のメモリ群b、203は第3のメモリ群c、204は第4のメモリ群dである。205は、メモリ群bに与える上位アドレスのラッチ回路b、206はメモリ群cに与えるラッチ回路c、207はメモリ群dに与えるラッチ回路dである。208は、メモリ制御回路28のアドレスバス(図3の45に対応)、209はアドレスバス208の下位アドレスである。そのアドレスビット数は、フラッシュメモリチップのデータ消去最小単位あるいはファイル管理の最小単位の記憶容量をアクセスするのに相当するアドレスビット数とする。例えば、

データ消去最小単位が512バイトであれば、9ビットとなる。フラッシュメモリチップのデータ消去最小単位がファイル管理の最小単位よりも小さい場合に限り、ファイル管理の最小単位に相当するアドレスビット数とすることが有効である。210は、アドレスバス208の下位アドレス209を除いた上位アドレスである。但し、メモリ群をアクセスするのに必要なさらに上位のアドレスは除くものとする。211は、メモリ群bをアクセスするためにアドレスラッチbに記憶された上位アドレス、212はメモリ群cをアクセスするためにアドレスラッチcに記憶された上位アドレス、213はメモリ群dをアクセスするためにアドレスラッチdに記憶された上位アドレスである。214はメモリ群a201のメモリ制御信号、215はメモリ群b202のメモリ制御信号、216はメモリ群c203のメモリ制御信号、217はメモリ群d204のメモリ制御信号である。

【0051】図9の構成において、メモリ群aからメモリ群dまでを同時にアクセスする場合には予めアドレスラッチ205~207にそれぞれのメモリ群をアクセスする上位アドレスを書き込んでおく。そして、アクセスを行う際には、下位アドレス210はすべてのメモリ群に共通であるため、アドレスバスはメモリ群aだけをアクセスするためのアドレスを供給し、メモリ群a以外のメモリ群は対応するアドレスラッチ回路から上位アドレスによりアクセスを行う。なお、アクセスの制御はメモリ制御信号214~217に委ねられ、例えば、メモリ群aとメモリ群bだけのアクセスであれば、メモリ制御信号214、215だけがアクティブになるようにする。また、例えばメモリ群dだけのアクセスにおいては、メモリ群aにはメモリ制御信号214をアクティブ

15

い。以上により、各メモリ群に異なるアドレスを与えることが可能となり、メモリ群さえ4つに分散していれば、物理的にアドレスの異なる箇所格納されている同一ファイルのデータを同時にアクセスすることができ、高速化の貢献度を高めることができる。図9の構成では、メモリ群aのアドレスラッチを省略することができる。勿論、メモリ群aにもアドレスラッチを設けることが有効であれば、そのようにしてもよい。

【0052】以上の実施例によれば、CPU1は比較的小さいレジスタに簡単な指定を行うことで、所望のセクタのリードライトアクセスを高速にアクセスできる。また、ファイルコントローラをワンチップマイコンとしてきめ細かい制御をソフトウェアにより指示することができ、DMA転送制御回路を搭載することによりワンチップマイコンがシステムより遅い動作しかできなくてもデータ転送を高速に行える。ただし、もしワンチップマイコンの動作速度がシステムに対して十分に高速な動作が可能であるならば、DMA制御回路は必要なく全てのデータ転送をワンチップマイコンが行う構成も考えられる。

【0053】また、本実施例ではデータバスをシステム32ビット、フラッシュメモリ8ビットとして説明を行ってきたが、16ビット動作のCPU1や64ビット動作のCPU、また16ビット入出力のフラッシュメモリなどに対しても、データラッチの数やデータ制御設定レジスタなどの構成やコントローラの制御プログラムなどを変えることにより容易に他のデータ幅に対応できる。その具体例を図15に示した。図15は各ビット幅の組合せにおける、ハード構成の具体的数値を示している。図の横の構成はフラッシュメモリのデータのビット数であり、4ビット、8ビット、16ビット、32ビットを例に挙げている。縦の構成はシステムのデータ幅を示しており、簡単な情報機器の8ビットから、現在のパソコンの主流である16ビット、32ビット、そして将来的なパソコンや高性能コンピュータの64ビット、128ビットの例を示している。またハード構成として挙げているデータ分配回路は、実施例のデータ分配回路61で説明すれば、図中の円一つのビット数×円の数で示している。各円の接続は図12～図14と同様の考え方で示すことができる。ラッチ数は実施例におけるラッチ51～54の数の増減であるが、システムデータ幅よりメモリデータビット数の方が大きい場合にはラッチの位置はデータ分配回路メモリ側に構成されることになる。メモリ群数は実施例では4つに分けているメモリの分割数を示している。

【0054】ただし、本実施例を含めデータ分配回路の分配数やラッチ数、メモリ群数は最低限度の数であり、回路規模、端子数などに余裕があれば、増やすことによりさらに効果的なシステム構成となる。というのはメモリ群数を増やすことによりデータの格納メモリの選択に

16

余裕ができるため、先に述べたようなメモリの使用量、使用頻度の偏りをさらに防ぐことができる。また並列アクセスするメモリ数を増やすことができるため、高速化をさらに進めることができる。特にシステムデータ幅とメモリデータビット数が等しいかあるいはメモリのデータビット数の方が多い場合には、実施例で述べてきたようなデータバス幅を合わせるという効果は全くないが、複数のメモリチップを同時アクセスしていわゆるインターリーブ方式による高速化を実現することができる。この際にはメモリ群数を増やすことにより同時にデータ分配回路におけるメモリのデータの分配数や、ラッチ数を同時に増やす必要がある。

【0055】なお、本実施例ではフラッシュメモリがシステムからのデータ転送に追従できることとして説明しているが、そのためには、フラッシュメモリ内にライトバッファを搭載することが望ましい。今後、高速に書き込みを行えるフラッシュメモリが実現すれば、このようなライトバッファは必要なくなる。フラッシュメモリにライトバッファが内蔵されていない場合には、ライトバッファをフラッシュメモリとデータ制御回路の間に設置することができる。そしてライトアクセスの際には直接フラッシュメモリに書き込まずライトバッファに書き込みを行ってシステムからのデータ転送終了後にライトバッファからフラッシュメモリへの書き込みを行うことになる。この場合の実施例の構成図を図4に示した。

【0056】図4中、既出の番号はこれまでに説明したものと同じのものである。新出のものとして71はアクセスがリードかライトかによってデータ分配回路61との接続を切り換えるデータセレクト、72はライトデータを一時的に格納する32ビット幅のライトバッファ、73はデータセレクト71を切り換えるためのリードライト信号であり、コマンドレジスタ25のデータの一部を入力すると良い。もしアクセスがライトであればデータ分配回路61をライトバッファに接続し、ライトデータをライトバッファに格納してシステムからのデータ転送後コントローラがフラッシュへの書き込みを行う。リードであればデータ分配回路61がフラッシュメモリに直接接続され前述の図3のリードアクセスと同様の動作が実行される。以上の構成にすることにより、ライトバッファを内蔵しておらず、書き込みの遅いフラッシュメモリを使用してもシステムから見れば高速なアクセスが可能となる。

【0057】次にファイル管理の実施例として図5を例にして説明する。これまではハード構成の実施例を説明してきたが、實際上、フラッシュメモリにどのように格納するかについて以下に述べる。基本的にこの操作はシステムのCPU及びシステムプログラムとフラッシュメモリファイルのコントローラが行うことになり、ソフトウェアによる操作が中心となる。図5による説明ではそのソフトウェア自体の内容ではなく、その操作の結果と

して、フラッシュメモリの各メモリ群の単位容量エリア（1セクタ分）が総体的にどのようにファイルのセクタに割り当てられるかを示す。ファイルの各セクタに対して、順次のメモリ群に単位容量エリアが確保される。4以上の連続セクタに対しては、4セクタ（例えば、図の1-1~1-4）に対して、順次メモリ群1~4の1単位容量エリアが割り付けられる。但し、実際にはこれらの各セクタのデータは4つの単位容量エリア内に分散して格納される。2セクタ（例えば図の1-5~1-6）に対しては、2つのメモリ群の単位容量エリアが割り当てられる。この場合も各セクタのデータは2つの単位容量エリア内に分散して格納される。ファイル1の最後のセクタ1-7には単独のメモリ群の単位容量エリアが割り当てられる。

【0058】ここで、2セクタおよび4セクタの分散格納の様子について、図10および図11によりさらに詳細に説明する。図10は4バイト同時書き込みの例を示し、図11は2バイト同時書き込みの例を示す。前述と同様、ホストのシステムバスが32ビット、1メモリ群のメモリアクセス幅が8ビットとする。

【0059】図10において、メモリ群a~dへの4セクタ同時書き込みの例を示す。図中、211は、ホストのシステムバスからのデータを示し、32ビット（4バイト）を単位として送られてくる。222~225は、8ビット（1バイト）単位のデータの通し番号を示しており、222は1バイト目のデータ、223は2バイト目のデータ、224は3バイト目、225は4バイト目のデータである。以下、連続して4セクタ分として2048バイト目までが送られてくる。実際には、32ビットバスであるため、1バイト目から4バイト目までが同時に受領され、以下も同様である。51~54は図3で示したデータ一時記憶のためのデータラッチである。

【0060】4バイトずつの同時書き込みであるため、32ビットデータつまり4バイトを同時にデータラッチ51~54に格納した後、各バイトがメモリ群a~dに分散して1バイトずつ書き込まれる。次の32ビットデータについて各バイトがメモリ群a~dの後続部分に書き込まれる。このようにして、1セクタ分の512バイト目までが書き込まれると、続いて第2のセクタについて同様に4バイトずつメモリ群a~dの後続部分に書き込まれる。図5では、便宜上、各セクタを特定のメモリ群に割り当てるように図示したが、以上の説明のように、実際には、各セクタのデータは特定のメモリ群のみに格納されるのではなく、メモリ群a~dに分散して書き込まれる。ただし、4セクタ全体のデータがメモリ群a~dの4セクタ分のエリアに格納されることは事実である。

【0061】図11においては、メモリ群aとメモリ群bへの2バイト同時書き込みの例を示す。

【0062】2バイトずつの同時書き込みであるため、3

2ビットデータつまり4バイトを同時にデータラッチ51~54に格納した後、これが2バイトずつに二分され、前半の2バイトがまずメモリ群aとメモリ群bに1バイトずつ書き込まれ、次に後半の2バイトがメモリ群aとメモリ群bに1バイトずつ書き込まれる。このようにして、1セクタ分の512バイト目までが書き込まれると、続いて第2のセクタの1バイト目から同様にしてメモリ群a、bの後続部分に書き込まれる。この場合も、第1セクタは特定のメモリ群のみに格納されるのではなく、メモリ群a、bに分散して書き込まれ、同様に、第2セクタも第1セクタに続いてメモリ群a、bに分散して書き込まれる。

【0063】単一セクタの書き込みについては図示しないが、この場合には、同時にデータラッチ51~54の各バイトデータが順次あるメモリ群に格納される。この場合のみ1セクタのデータは分散されずに特定のメモリ群内に格納される。

【0064】以上から分かるように、重要なことは、以上の動作の結果、フラッシュメモリ素子の最小消去単位（ここでは1セクタ容量に相当する512バイト）のエリア内には異なるファイルのデータが混在しないことである。これが保証されることにより、ファイル書き時に別のファイルのデータを消去することなく、かつ、メモリの記憶エリアを有効に利用しながら複数バイト並列同時書き込みにより高速のファイルデータ格納を達成することが可能になる。

【0065】前述のように、図5（1）はある程度のファイルの書き込みが行われた状態、図5（2）はその後あるファイルについては更新されかつファイル容量が増えてしまった状態を示している。図5（1）では格納されるファイル番号順、セクタ番号順に異なるメモリが割り当てられるように格納する。管理上無駄がないように、原則として、詰めて空きがないようにする。

【0066】図5（1）に示したように、メモリ群に空きが生じないようにファイルのセクタを、順次、連続的に詰めて格納するものとする、ファイル更新によりファイル容量が増えた場合には図5（2）に示すように物理的な格納位置としては連続的に格納することができない場合が生じる。このような場合であっても、メモリ群の順番としては連続的になるように格納する。つまり、ファイル番号4番のファイルを例にすると、（1）では4-1から4-5までの5セクタが格納されているが、（2）で2セクタ追加されると、4-1がメモリ群3から開始されているため、4-6、4-7の増加分はメモリ群2、3のエリアが確保されることになる。セクタ数が増加したファイルの書き込み時には、この時には4-5と4-6と2セクタ連続セクタとして扱われ、4-7は単独セクタとして扱われることになる。つまり追加セクタに対しても既格納セクタと合わせて連続セクタとしての扱いを受けることになる。これらのファイル管理の格

19

納と読み出しをスムーズに行うためには、ハードウェア構成としてシステムからのアクセスセクタの指定は開始セクタとセクタ数にし、物理的な格納位置の把握はファイルシステム内部において行われた方が適当であると考えられる。つまりシステムからのアクセスファイルの指定は高速化のため極力簡略化すべきであるが、物理的な位置が分散していると物理的な位置の指定が複雑化してしまう。従って図2に示したエンドセクタ24はこの意味からは取り去るべきである。そしてファイルの連鎖を示す情報をメモリにデータ以外の情報を格納する冗長領域があればそこに、なければ他の記憶手段に格納することにより、ファイルの開始番号が指定されればそれに続くセクタ番号の物理的な位置が全て連鎖して明らかになるため、連続アクセスが可能となる。

【0067】図6には連鎖情報の例を示している。図中、85はメモリに格納されているファイルデータであり、ファイル番号4、セクタ番号5のデータである。86はそれに続くファイル番号4、セクタ番号6のデータである。そして87はさらにそれに続くファイル番号4、セクタ番号7のデータである。ただし、これらのデータはセクタ番号を順番にしているが、実際の格納状態としては混じり合っている場合がある。つまり、連続セクタアクセスにより複数チップ同時アクセスすることによりデータをスクランブルして格納している場合がある。もっともこのときにもバイトごとのデータ格納の順番付けが必要であるため、セクタ番号は非常に重要である。88はファイルデータ85が格納されているメモリ上の物理的な位置を示す物理アドレスであり、左側の数字"3"はメモリ群番号、右側の数字"5"はそのメモリ群の中のアドレスである。89、90は同様にファイルデータ86、ファイルデータ87の物理アドレスである。91はファイルデータ85の次のセクタが格納されている物理アドレスを示す連鎖情報であり、ファイルデータ85がファイル番号4、セクタ番号5のデータであるため、連鎖情報91はファイル番号4、セクタ番号6のデータが格納されている物理アドレスを示していることになる。左側の数字がメモリ群番号、右側がメモリ群の中のアドレスである。同様に、連鎖情報92はファイル番号4、セクタ番号7のデータが格納されている物理アドレスを示しており、連鎖情報93は次のセクタが存在しない場合を示している。つまりファイル番号4のファイルは7セクタで終了することがわかる。この連鎖情報がファイルデータに添付されていれば、システムのCPUはファイル格納の物理的な位置を把握する必要はなく単にファイル自体をアクセスする形式をとれる。そしてファイル制御のコントローラが連鎖情報を参照しながらメモリの物理的なアクセスを行うことにより、一つのファイルでありながら物理的な格納位置が連続しておらず分散しているファイルでも連続的なアクセスが可能となる。その際、異なるメモリ群の同時アクセスのために

20

異なるアドレスを入力する構成が必要である。ただし、図9で前述したように、1セクタアクセス分の下位アドレスは共用でき、それ以上の上位アドレスに関してだけで十分である。以上、連鎖情報を用いる実施例によればシステムからのアクセス指定を簡略化でき、少ない情報量で実施可能である。

【0068】次にメモリ素子自体に各種機能を組み込む実施例について説明する。

【0069】図7は複数のメモリチップを1つのパッケージに収め、それに付加回路を設けたメモリ素子の構成図である。図中、101はメモリパッケージ、102から105は同様の機能を持ったメモリチップ、106から109はメモリパッケージ101の入出力データ端子で、それぞれ一つで一つのメモリチップのデータ入出力が可能である。例えばメモリチップ102が8ビットのデータ入出力であれば106から109まで全部で8ビット×4=32ビットのデータ入出力端子を持つことになる。110はこれらのメモリチップと入出力端子を接続する経路を指定する信号入力端子、111は指定信号、112は接続経路設定回路である。本メモリ101のユーザは、データ入出力端子106から109の設定を信号入力端子110により行ってからアクセスを実行する。この設定には先述の図12、図13、図14のような接続経路から選択するものが一例として考えられる。メモリ101では設定されたデータ接続を指定信号111として接続経路設定回路112に送り、接続設定回路112では設定に従ってメモリチップ102から105をデータ入出力端子106から109に接続し、所望のデータパスにメモリチップのデータを接続することを実現する。

【0070】図8は、ユーザからのデータ接続の設定をコマンド入力により行うメモリ素子の構成例である。図中、113はコマンド設定値のレジスタとその設定値から指定信号111を作り出すコマンド制御回路である。114はデータ入出力端子の一部を使用してユーザがコマンド設定値を設定するデータ線である。その他は図7の同じ番号と同一のものである。ユーザはデータの接続経路を図12、図13、図14を一例として選択し、コマンドコードとしてコマンド制御回路113にデータパスの一部114から設定し、コマンド制御回路113では設定されたコマンドコードから接続経路設定回路112に該当する指定信号111を送る。以下は、図7の説明と同様である。図7、図8による本実施例によれば、メモリ素子としてこれまでに説明した他の実施例を実現することができ、周辺回路の削減を図ることができる効果がある。なお本実施例ではメモリパッケージ内に複数のチップと制御回路を搭載することとしているが、これらを一チップにまとめることにより小型化と高速化を助長できる。

【0071】

21

【発明の効果】本発明によれば、フラッシュメモリ装置において、ファイル書替時に別のファイルのデータを消去することなく、かつ、メモリの記憶エリアを有効に利用しながら複数バイト並列同時書き込みにより高速のファイルデータ格納を達成することが可能になる。

【図面の簡単な説明】

【図1】本発明を実現するシステム構成を示すブロック図

【図2】フラッシュメモリファイル装置の実施例の構成を示すブロック図

【図3】実施例の動作を説明するための主要部分のブロック図

【図4】実施例における書き込みが速いフラッシュメモリチップを使用した場合のデータ制御を示す説明図

【図5】実施例におけるメモリへのファイル格納の実行例の説明図

【図6】本発明の他の実施例における連鎖情報格納の実行例の説明図

【図7】信号端子入力指定によるデータ分配機能をメモリ素子内に備えたメモリ構成例の説明図

【図8】コマンド設定入力指定によるデータ分配機能をメモリ素子内に備えたメモリ構成例の説明図

【図9】実施例におけるメモリ群のアドレス接続の説明図

【図10】実施例における4バイト同時書き込み時の動作の説明図

22

【図11】実施例における2バイト同時書き込み時の動作の説明図システムバス32ビット、

【図12】実施例におけるメモリ1チップ同時アクセスのデータ分配の説明図

【図13】実施例におけるメモリ2チップ同時アクセスのデータ分配の説明図

【図14】実施例におけるメモリ4チップ同時アクセスのデータ分配の説明図

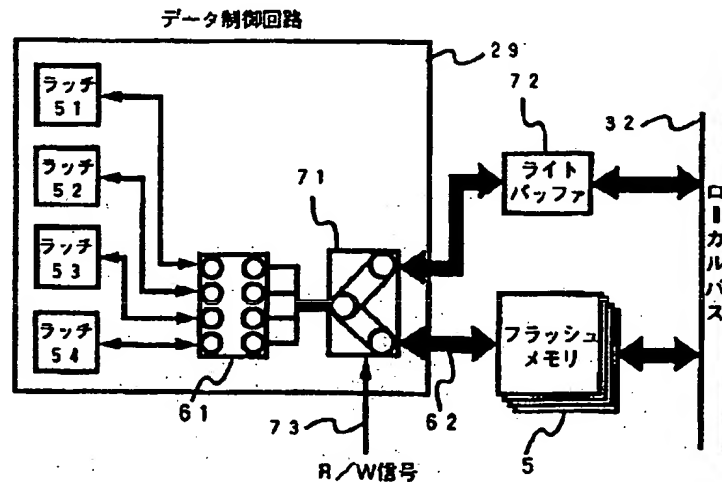
【図15】システムバスとメモリのデータバスの様々な構成におけるハード構成の説明図

【符号の説明】

2…クロック発生回路、4…ファイル制御回路、5…フラッシュメモリ、15…レディ制御回路、21…I/Fレジスタ、22…ステータスレジスタ、23…スタートセクタレジスタ、24…エンドセクタレジスタ、25…コマンドレジスタ、26…データレジスタ、27…コントローラ、28…メモリ制御回路、29…データ制御回路、30…DMA制御回路、31…状態提示信号、51…ラッチ回路、52…ラッチ回路、53…ラッチ回路、54…ラッチ回路、55…ラッチ信号発生回路、56…データ制御設定レジスタ、61…データ分配回路、71…データ切り換え回路、72…ライトバッファ、91…連鎖情報、101…メモリパッケージ、110…指定信号入力端子、112…接続経路設定回路、113…コマンド制御回路

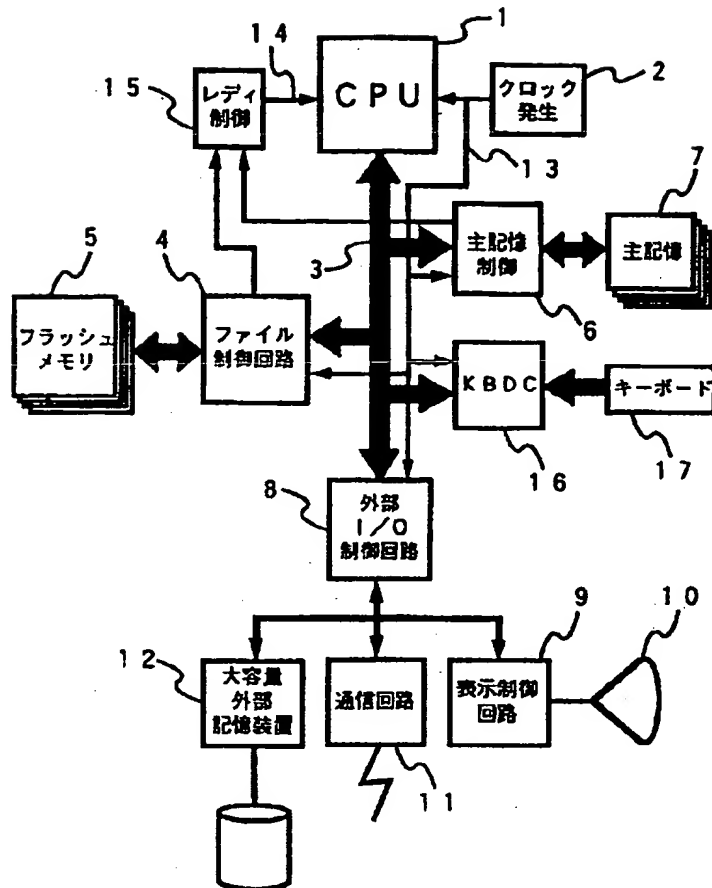
【図4】

図4



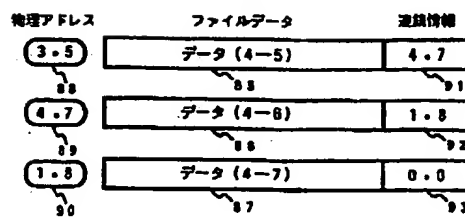
【図1】

図1



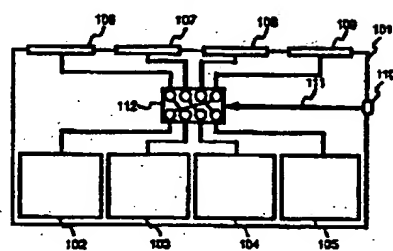
【図6】

図6



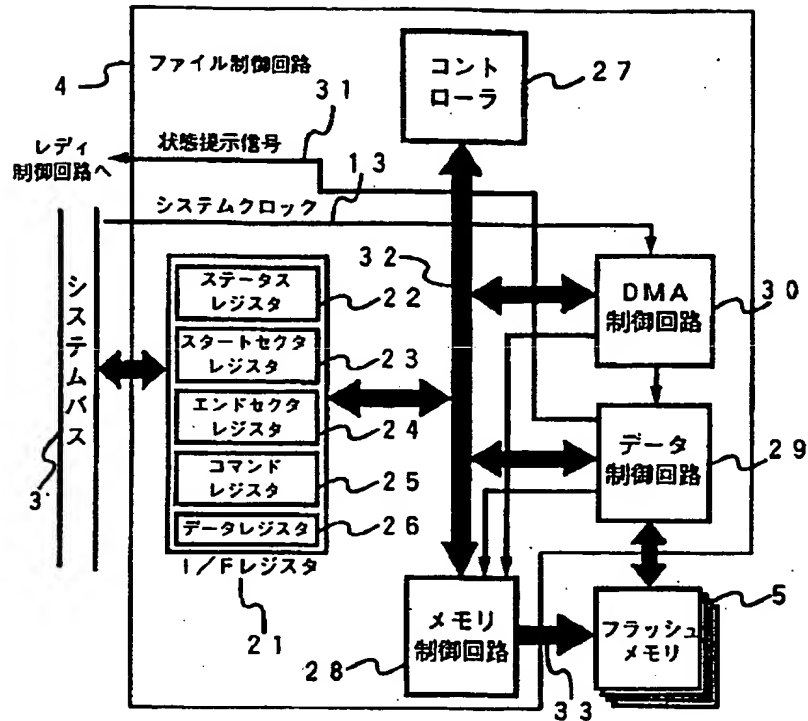
【図7】

図7



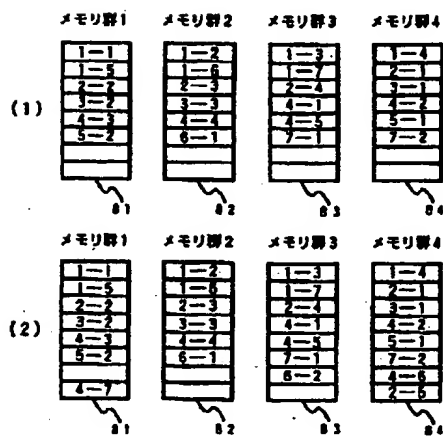
【図2】

図2



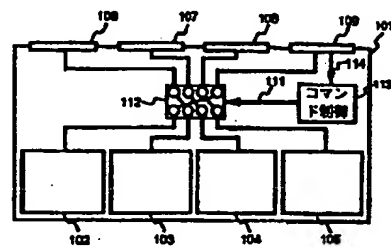
【図5】

図5



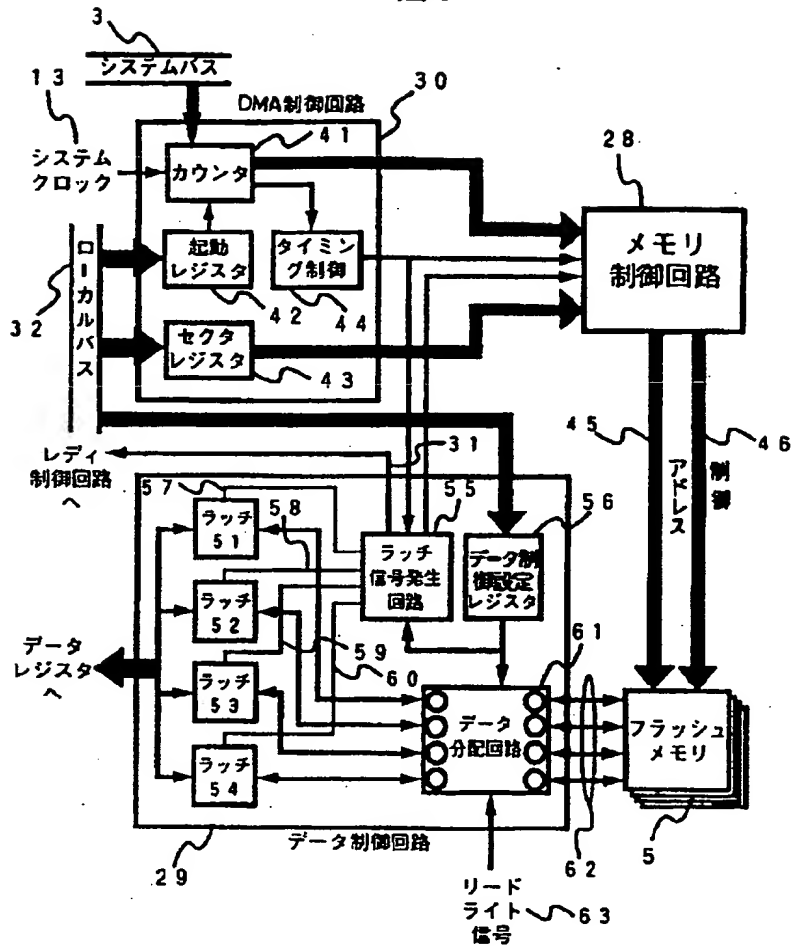
【図8】

図8

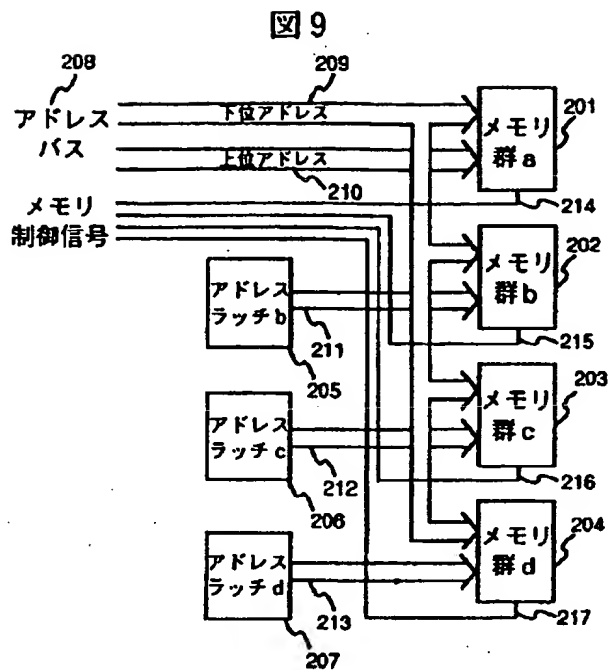


【図3】

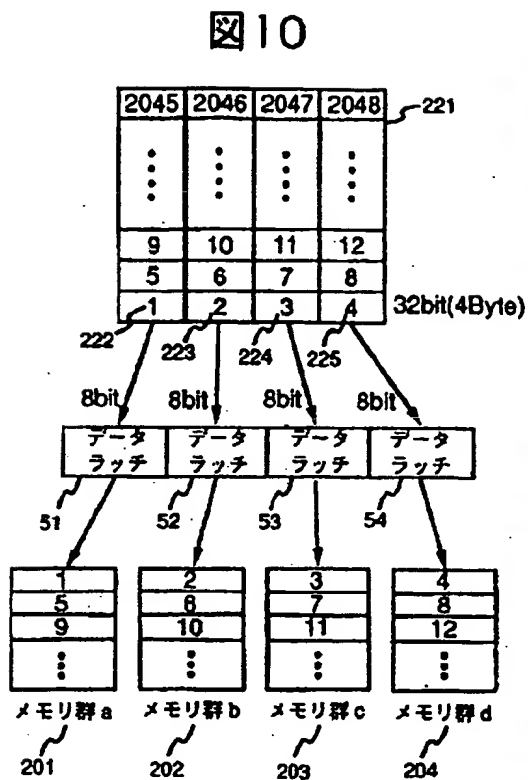
図 3



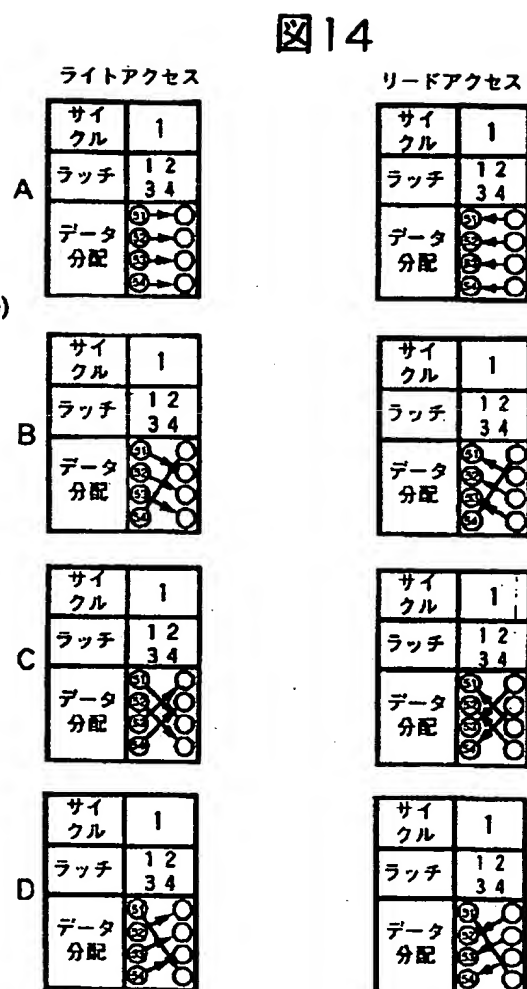
【図9】



【図10】

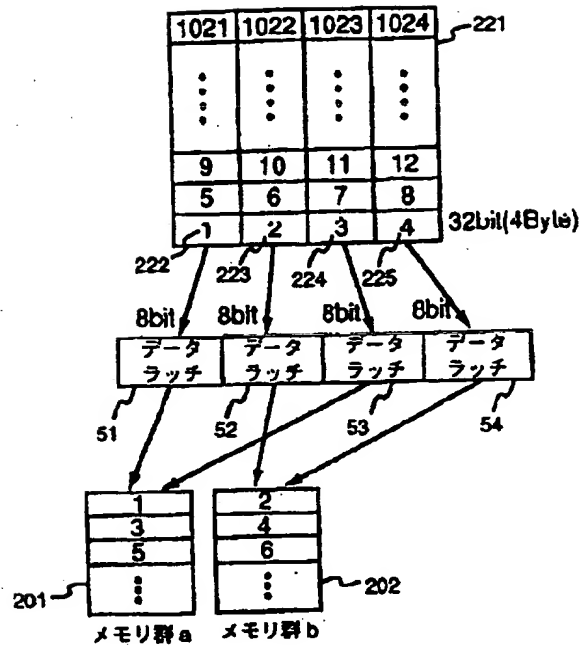


【図14】



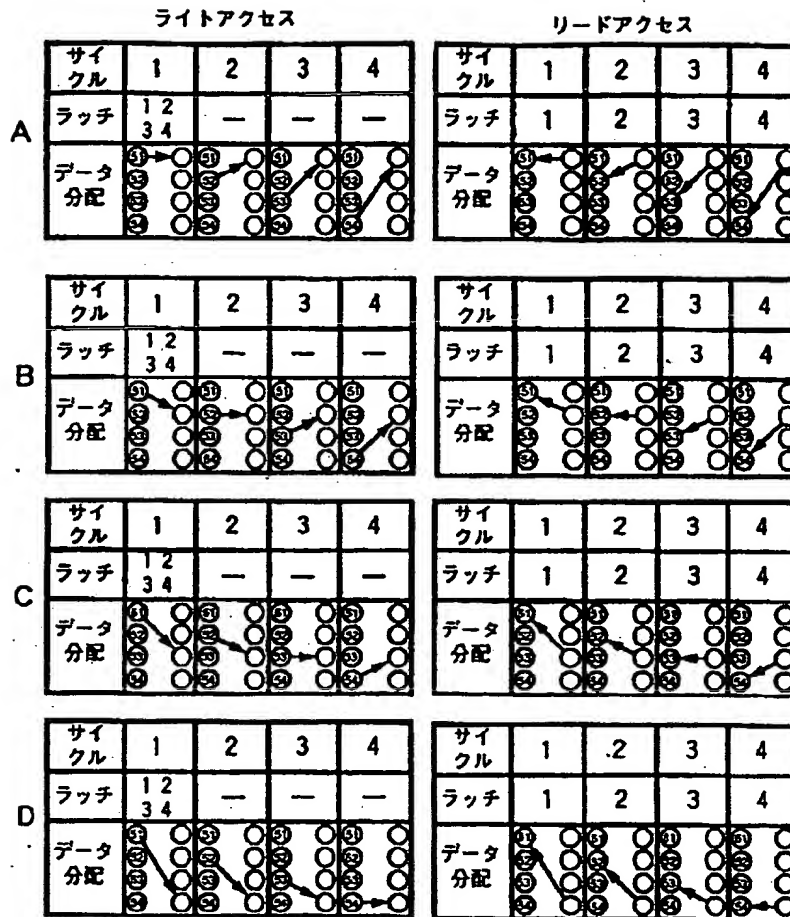
【図11】

図 11



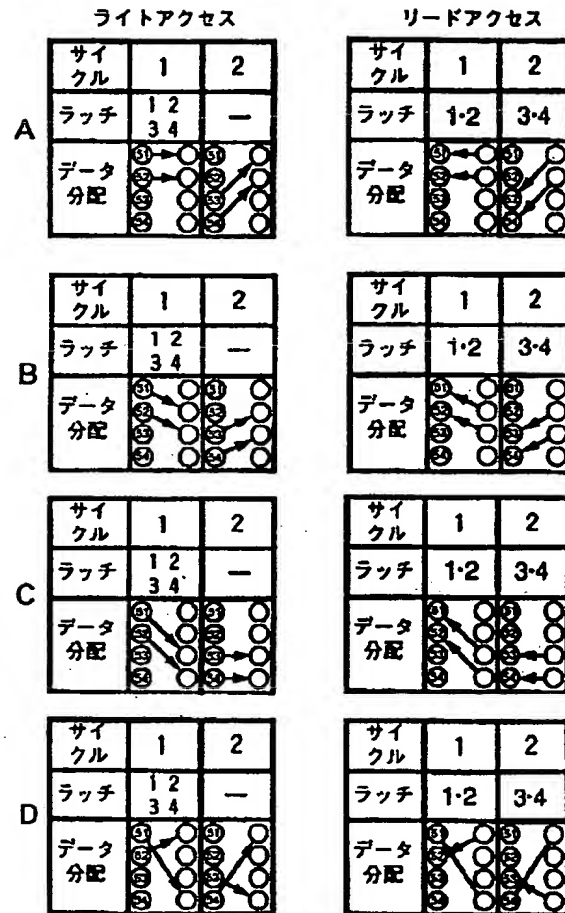
【図12】

図12



【図13】

図13



【図15】

図15

システム データ バス幅	メモリデータ ビット ハード構成	4 bit	8 bit	16 bit	32 bit
8 bit	データ分配回路	4 bit×2	8 bit×1	8 bit×2	8 bit×4
	ラッチ数	2	1	2	4
	メモリ群数	2	1	1	1
16 bit	データ分配回路	4 bit×4	8 bit×2	16 bit×1	16 bit×2
	ラッチ数	4	2	1	2
	メモリ群数	4	2	1	1
32 bit	データ分配回路	4 bit×8	8 bit×4	16 bit×2	32 bit×1
	ラッチ数	8	4	2	1
	メモリ群数	8	4	2	1
64 bit	データ分配回路	4 bit×16	8 bit×8	16 bit×4	32 bit×2
	ラッチ数	16	8	4	2
	メモリ群数	16	8	4	2
128 bit	データ分配回路	4 bit×32	8 bit×16	16 bit×8	32 bit×4
	ラッチ数	32	16	8	4
	メモリ群数	32	16	8	4

フロントページの続き

(72)発明者 北原 潤
 神奈川県横浜市戸塚区吉田町292番地 株
 式会社日立製作所マイクロエレクトロニク
 ス機器開発研究所内

(72)発明者 飛田 庸博
 神奈川県横浜市戸塚区吉田町292番地 株
 式会社日立製作所マイクロエレクトロニク
 ス機器開発研究所内

(72)発明者 古沢 和則
 東京都小平市上水本町五丁目20番1号 株
 式会社日立製作所半導体設計開発センタ内

Fig. 4

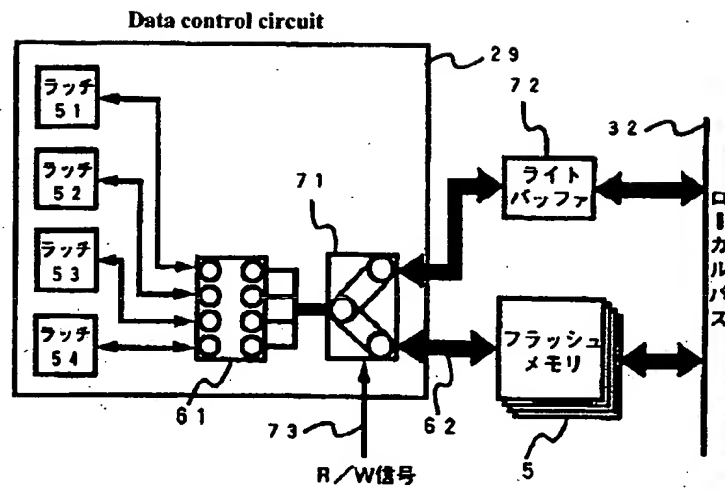


Fig. 1

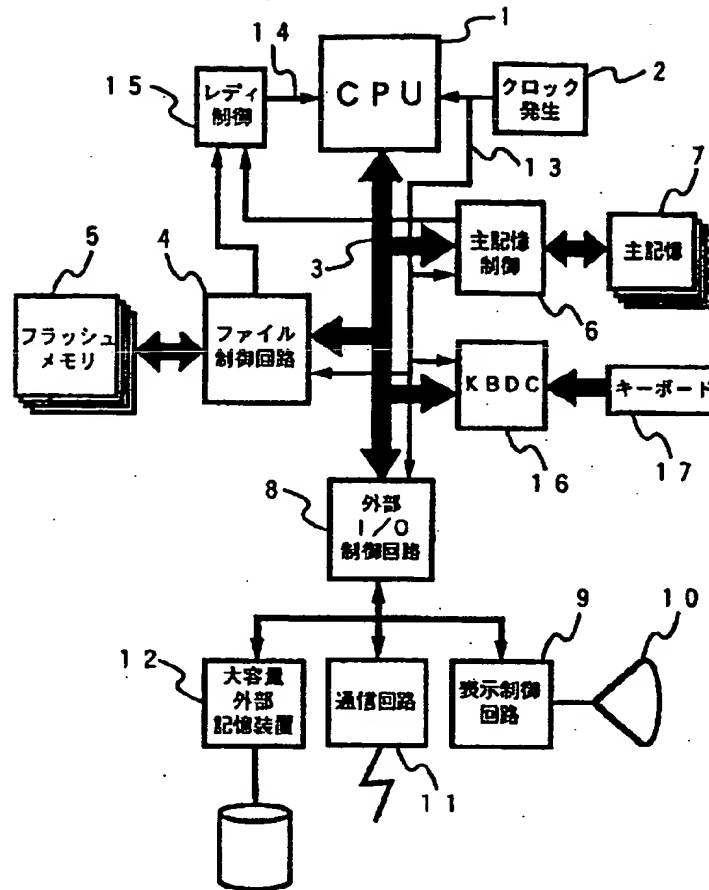


Fig. 6

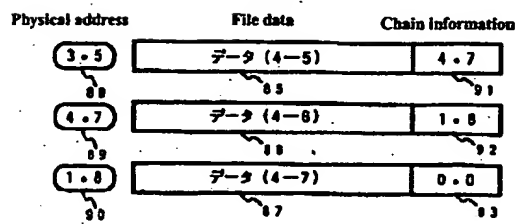


Fig. 7

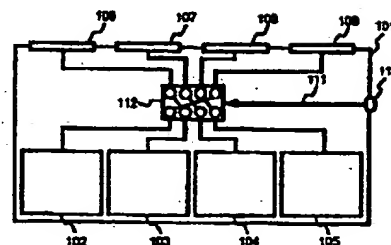


Fig. 2

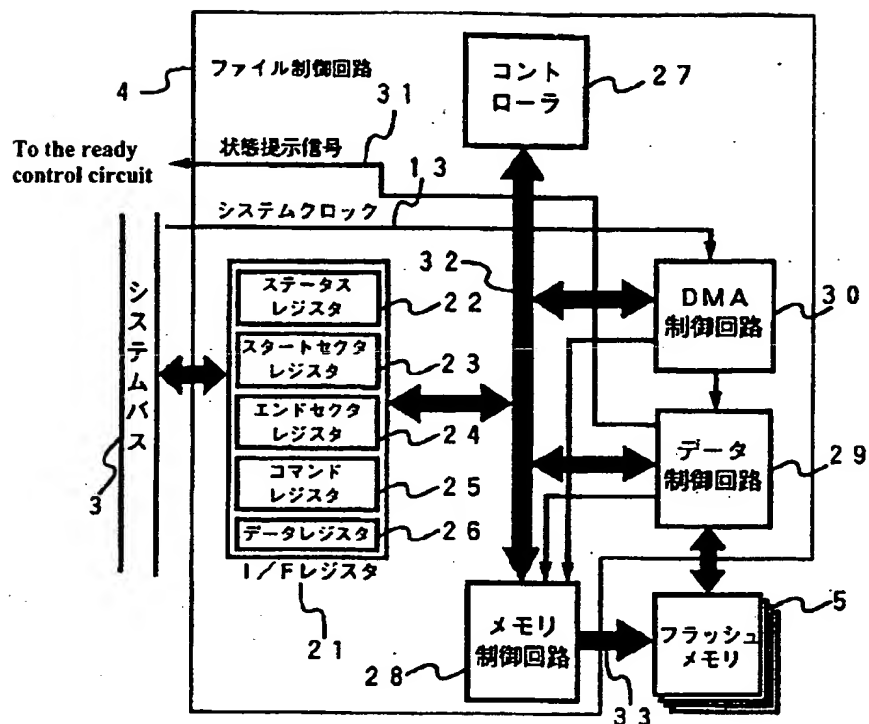


Fig. 5

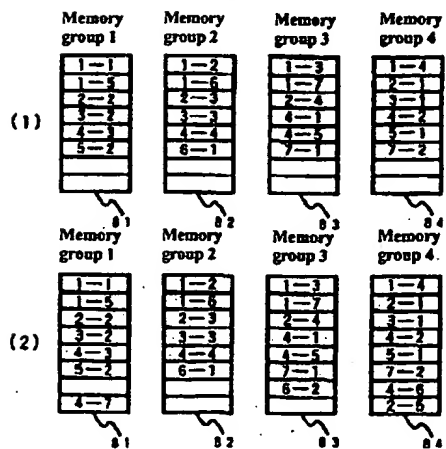


Fig. 8

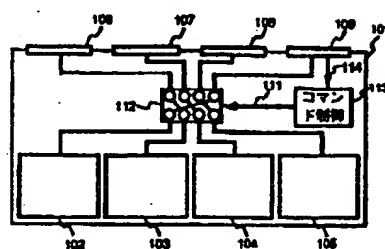


Fig. 3

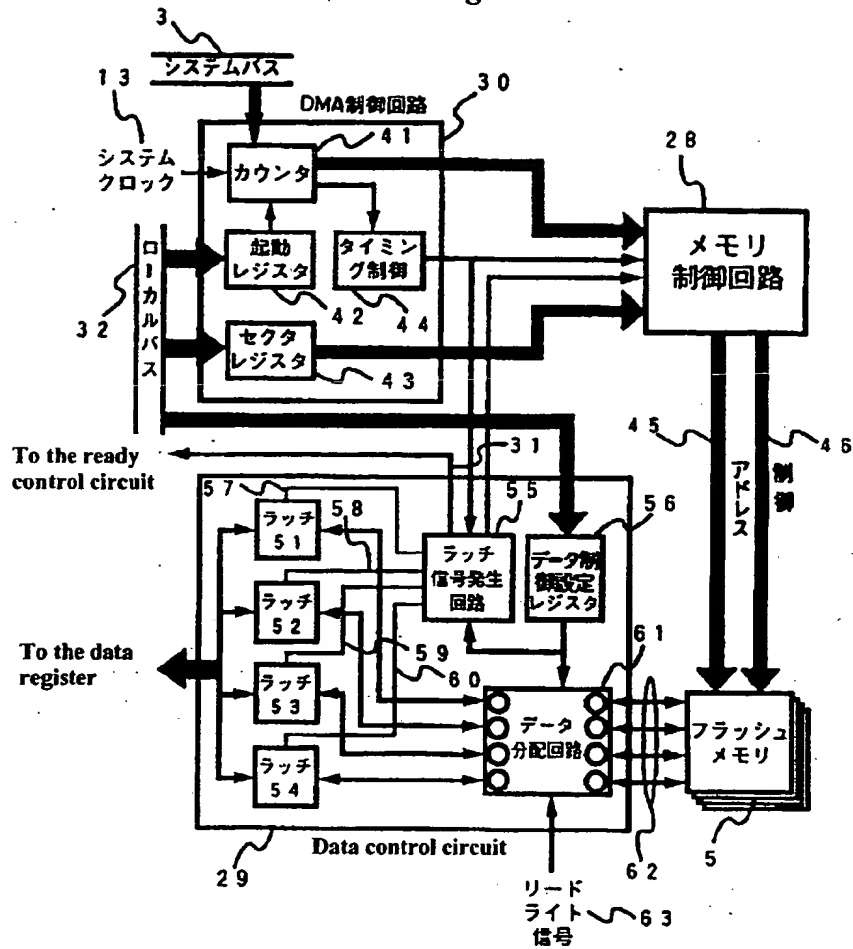


Fig. 9

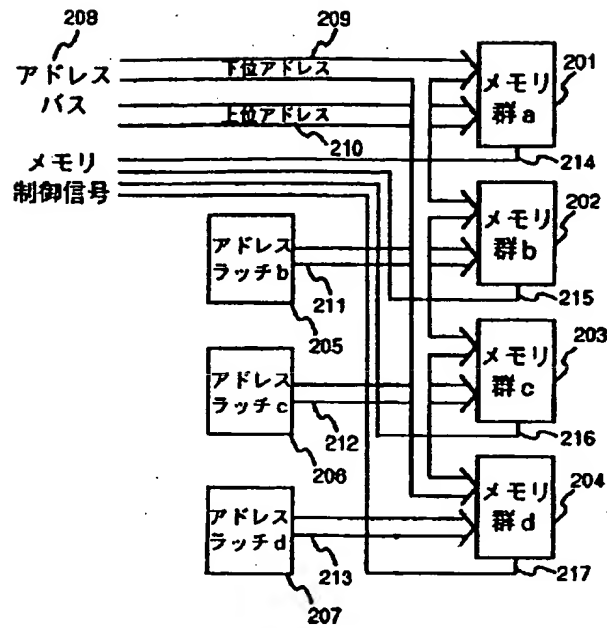


Fig. 10

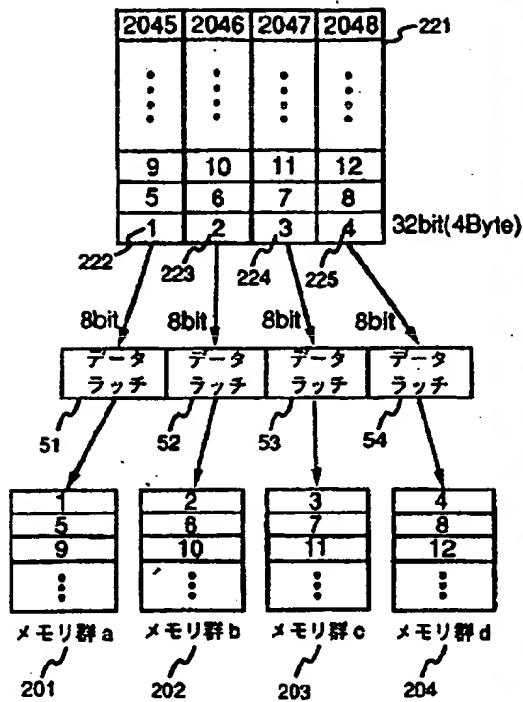


Fig. 14

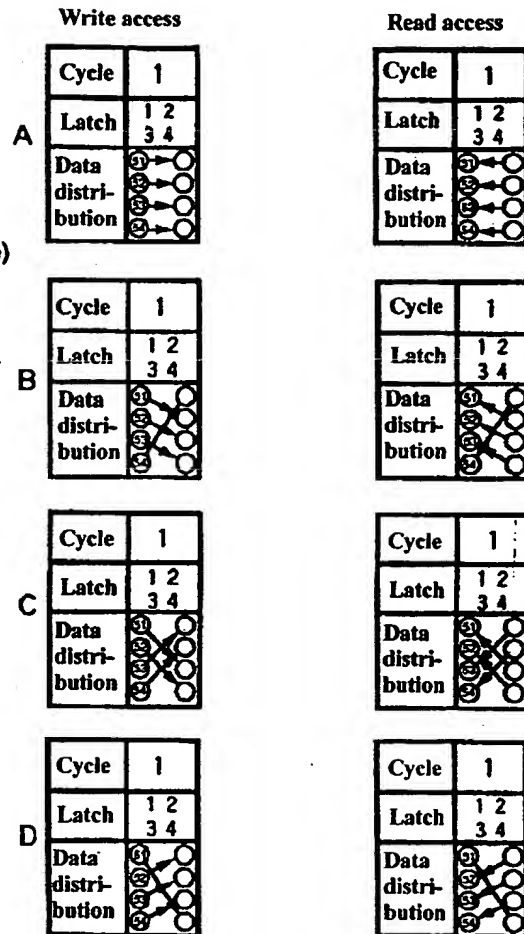


Fig. 11

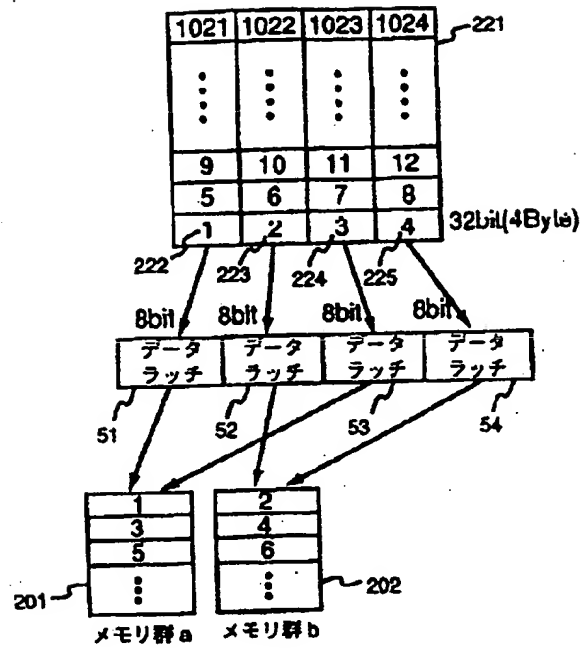


Fig. 12

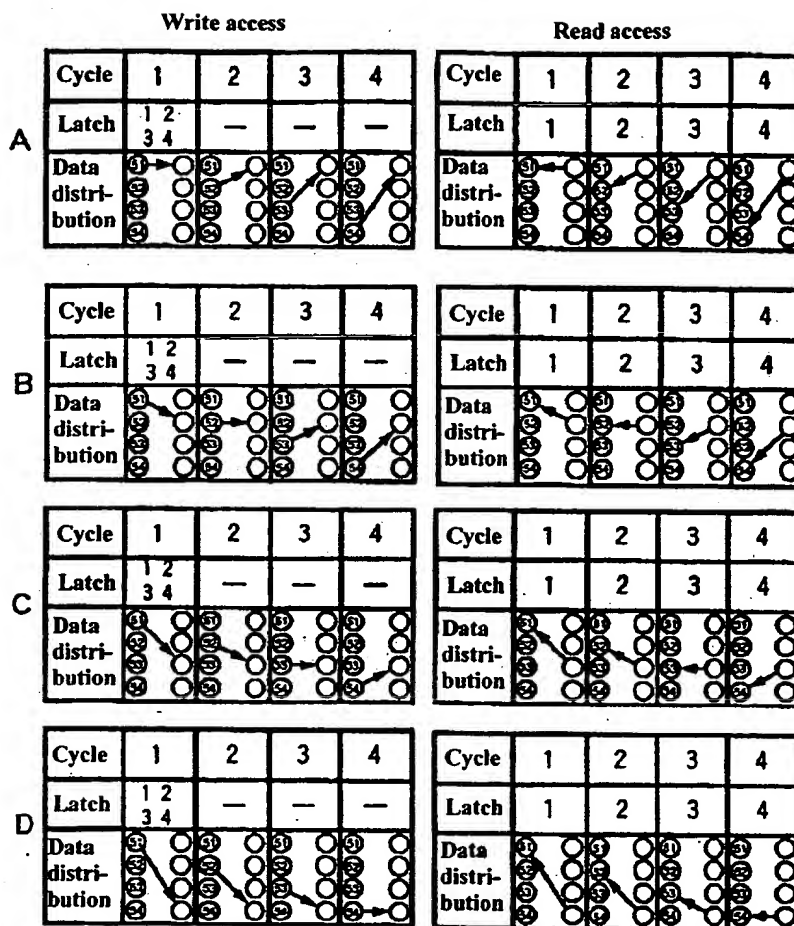


Fig. 13

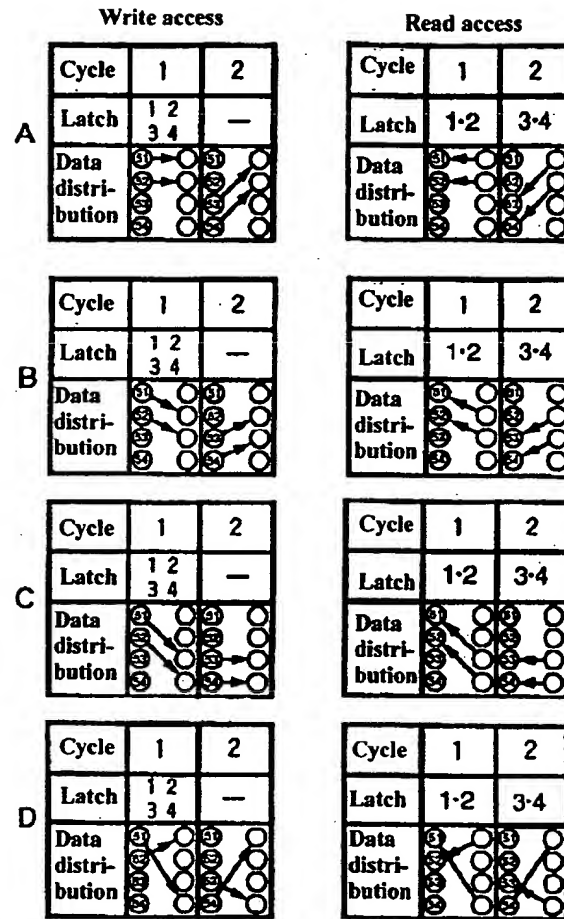


Fig. 15

System data bus width	Memory data bit Hardware configuration	4 bit	8 bit	16 bit	32 bit
8 bit	Data distribution circuit	4 bit \times 2	8 bit \times 1	8 bit \times 2	8 bit \times 4
	Number of latches	2	1	2	4
	Number of memory groups	2	1	1	1
16 bit	Data distribution circuit	4 bit \times 4	8 bit \times 2	16 bit \times 1	16 bit \times 2
	Number of latches	4	2	1	2
	Number of memory groups	4	2	1	1
32 bit	Data distribution circuit	4 bit \times 8	8 bit \times 4	16 bit \times 2	32 bit \times 1
	Number of latches	8	4	2	1
	Number of memory groups	8	4	2	1
64 bit	Data distribution circuit	4 bit \times 16	8 bit \times 8	16 bit \times 4	32 bit \times 2
	Number of latches	16	8	4	2
	Number of memory groups	16	8	4	2
128 bit	Data distribution circuit	4 bit \times 32	8 bit \times 16	16 bit \times 8	32 bit \times 4
	Number of latches	32	16	8	4
	Number of memory groups	32	16	8	4

(19) Japan Patent Office (JP) (12) Public Patent Announcement (A) (11) Patent Application Publication
Pat App Hei 6-266596

(43) Publication: September 22, 1994

(51) Int. Cl. ⁵	ID Code	Internal Cl. No.	FI	Technology indication section
G 06 F 12/00	520 P	8944-5B		
	550	9366-5B		
Examination request: Not requested		Number of patent Claims: 11 OL (Altogether 21 pages)		

(21) Application No.: Pat App Hei 5-51041

292, Yoshida-cho, Totsuka-ku,
Yokohama City, Kanagawa Pref.

(22) Application Date: March 11, 1993

(72) Inventor: Kenichi Kaki
 c/o Microelectronics Equipment
 Development Laboratory
 Hitachi, Ltd.
 292, Yoshida-cho, Totsuka-ku,
 Yokohama City, Kanagawa Pref.

(71) Applicant: 000005108
 Hitachi, Ltd.
 6, Surugadai 4-chome, Kanda,
 Chiyoda-ku, Tokyo

(72) Inventor: Kunihiro Katayama
 c/o Microelectronics Equipment
 Development Laboratory
 Hitachi, Ltd.

(74) Agent: Kazuko Tomita, Patent attorney

(Continued to the last page)

(54) [Title of the Invention] Flash memory file storage device and information processor.

(57) [Abstract] (Modifications exist.)

[Construction]

In a file storage device using flash memory elements whose minimum deleting unit is larger than X bits and access width is x bits ($x = X/p$; p is an integer equal to or greater than 2) through an X-bit data bus, it is equipped with a flash memory device 5 consisting of p sets of flash memory element groups accessible simultaneously, a dividing means 51~54 which divide data on the data bus into at least p parts, a data distributing means 61 which has at least a first function to make p pieces of x-bit data correspond to one set of the flash memory element groups and a second function to make each of the p pieces of x-bit data correspond to another set of the flash memory element groups, and a control means 56 which controls the data distribution means 61 so that the first and second functions are switched according to the number of storage capacity units for file management of the access target file.

[Efficacy]

It becomes possible to read multiple bytes simultaneously in parallel while effectively utilizing the memory area.

[Scope of Patent Claims]

[Claim 1]

In a flash memory file storage device which is a file storage device storing data through an X-bit data bus and which uses flash memory elements whose minimum deleting unit is larger than X bits and the access width is x bits ($x = X/p$: p is an integer equal to or greater than 2), a flash memory file storage device characterized by being equipped with a flash memory device consisting of p sets of flash memory element groups accessible simultaneously, a dividing means which divide data on the X-bit data bus into at least p parts, a data distributing means which has at least a first function to make p pieces of x-bit data obtained by the division correspond to one set of the flash memory element groups and a second function to make each of the p pieces of x-bit data correspond to another set of the flash memory element groups, and a control means which controls the data distribution means so that the first and second functions are switched according to the number of storage capacity units for file management of the access target file.

[Claim 2]

The flash memory file storage device described in Claim 1 characterized by the fact that the control means controls the data distribution means so that data of different files do not coexist in the memory area of the minimum deleting unit of the flash memory element.

[Claim 3]

The flash memory file storage device described in the Claim 1 or 2 characterized by the fact that the control means controls the data distribution means so that the second function is selected for data of continuous p pieces of storage capacity units in the file, and that the first function is selected for data of one storage capacity unit

[Claim 4]

The flash memory file storage device described in the Claim 1 or 2 characterized by the fact that the storage capacity unit and the minimum deleting unit of the flash memory element are equal to each other.

[Claim 5]

In an information processing device having as its components a central computing/processing means which processes programs and data, a clock oscillation means which drives the central computing/processing means, a file storage means having flash memory as its storage medium, a file storage control means which controls access of the flash memory of the file storage means, an information processing device loaded with a flash memory file storage device characterized by the fact that the same signal or synchronized signal with clock signal generated by the clock oscillation means is input to the file storage control means, and that the central computing/processing means and the file storage control means perform synchronized actions to send/receive file data.

[Claim 6]

The information processing device described in the Claim 5 characterized by the fact that when the one-time data access width is different between the central computing/processing means and the file storage means, the file storage control means is equipped with a means to output a status indication signal indicating whether file data are being processed or the processing is complete, the central computing/processing means has a function wherein it receives the status indication signal, if it indicates that it is in process, it halts the process until the processing is complete, the file storage control means has a data bit width control means which adjust the number of data bits dealt with by the file storage means to the processing data bit width of the central computing/processing device, the data bit width control means shows to the central computing/processing means that processing is taking place using the status indication signal, and allows it to stand by during the period necessary for adjusting the data bit widths, and performs data exchange by adjusting the processing data width of both.

[Claim 7]

The information processing device described in the Claim 6 characterized by the fact that the file storage means is composed of the number of flash memory groups necessary for generating data of the same number of bits with the data access width of the central computing/processing means, wherein sequential order is regulated for the memory groups, storing files over multiple storage capacity units following this regulated order of memory groups, and when the number of storage capacity units increases by updating a once-stored file, the increase is secured from the memory group regulated in order as the next of memory group where the last data of the file were stored in the previous storing.

[Claim 8]

In an information processing device equipped with a central processing device and a file storage device, characterized by the fact that multiple flash memory elements are used as the storage media of the file storage device, where the minimum deleting unit of data of the flash memory is made to be equal to the storage capacity unit for file management, and if the number of storage capacity units when the central processing device requests file access is plural, the multiple flash memory elements are accessed simultaneously, and if the number of storage capacity units is one, one piece of the multiple flash memory elements is accessed.

[Claim 9]

In a file storage device having flash memory as its storage medium, a flash memory file storage device characterized by the fact that if the storage capacity of a file to record extends over multiple storage capacity units, it is equipped with a chaining information storage means which records information in the physical location where the next storage capacity unit is stored for each storage capacity unit.

[Claim 10]

In a memory element where multiple memory chips are embedded in a package, a memory element where the memory package is equipped with the corresponding number of input/output data terminals to the total number of data inputs/outputs of all the memory chips, equipped with a data control means which controls switching/connecting input/output data of each memory chip and data input/output terminals and a control signal terminal for instructions from a memory user to the data control means, making it possible to switch/connect data of the multiple memory chips upon instructions of the control signal terminal to connect input/output data of an arbitrary memory chip to an arbitrary terminal of the input/output data terminals.

[Claim 11]

In a memory element where multiple memory chips are embedded in a package, a memory element where the memory package is equipped with the corresponding number of input/output data terminals to the total number of data inputs/outputs of all the memory chips, equipped with a data control means which controls switching/connecting input/output data of each memory chip and data input/output terminals and a control command setting means for instructions from a memory user to the data control means, making it possible to switch/connect data of the multiple memory chips upon instructions of the control command setting means to connect input/output data of an arbitrary memory chip to an arbitrary terminal of the input/output data terminals.

[Detailed Explanation of the Invention]**[0001]****[Field of Industrial Application]**

The present invention relates to information equipment loaded with a file storage device having flash memory as its storage medium, and especially relates to a file access method of file storage devices suitable for information equipment where high speed file access is demanded.

FH 008711

[0002]

[Prior Art Technology]

In today's general information equipment starting with personal computers, auxiliary storage devices are almost indispensable peripheral equipment. Moreover, in general, auxiliary storage devices are contained in most equipment, and arranged so that the users can always deal with files of large capacities.

[0003]

Recently, there are increasing cases where personal computers of notebook size and palm-top type are used, and portability is coming to be regarded important. Therefore, in place of magnetic disk storage devices which are weak in terms of vibration and have a large power consumption, file storage devices having semiconductors as their storage media are drawing attention. For example, disclosed in Pat Open Hei 2-292798 Public Report is a technology of a semiconductor file storage device using a flash memory as its storage medium.

[0004]

Flash memory is nonvolatile memory which can be electrically rewritten, and because capacity increase and cost reduction are possible, it is one of the most effective memories as storage media of semiconductor file storage devices. The technology in the Pat Open Hei Public Report uses this memory to solve the problems in constructing a file storage device and devices for improving the ease of use. For example, it proposes a method of overcoming the shortcoming of flash memory in which the elements become degraded by frequent rewriting, a method of increasing the speed in a phase called, deleting necessary for rewriting a flash memory, etc. Also, as an interface with information equipment which becomes a host, it proposes it to be the same as that for magnetic disk devices, directed towards the construction of systems that can replace magnetic disk devices.

[0005]

[Problems overcome by the Invention]

The semiconductor file storage device of prior art technology uses the pre-existing interface bus of the host information equipment and places importance on compatibility with magnetic disk devices. Although this makes the semiconductor file storage device to be easily accepted by the users, in return for placing importance on compatibility with magnetic disk devices, no consideration is made on superiority of semiconductor storage elements to magnetic disk devices.

[0006]

Namely, although semiconductor storage elements enable very high speed data access because they are static recording media unlike magnetic disk devices which pick up data from rotating disks and perform writing, there is the problem that the superiority of this high speed access cannot be utilized with the same interface as with magnetic storage devices.

[0007]

It is an objective of the present invention to construct an interface which achieves high-speed access performance to the maximum by making semiconductor storage elements to be its storage media and to provide new auxiliary storage devices which improve the access performance, which is a shortcoming of magnetic disk devices.

[0008]

[Problem Resolution Means]

The flash memory file storage device by this invention is characterized by the fact that in a flash memory file storage device which is a file storage device storing data through an X-bit data bus and uses flash memory elements

FH 008712

whose minimum deleting unit is larger than the X bits and access width is x -bits ($x = X/p$: p is an integer equal to or greater than 2), it is equipped with a flash memory device consisting of p sets of flash memory element groups accessible simultaneously, dividing means which divide data on the X -bit data bus into at least p parts, a data distributing means which has at least a first function to make p pieces of x -bit data obtained by the division correspond to one set of the flash memory element groups and a second function to make each of the p pieces of x -bit data correspond to another set of the flash memory element groups, and a control means which controls the data distribution means so that the first and second functions are switched according to the number of storage capacity units for file management of the access target file.

[0009]

[Operation]

In magnetic disk devices used in the current person computers, data access is slower than the memory access of the personal computer and does not have to operate in synchronization with the CPU in the personal computer. Therefore, the give and take of data of magnetic disk devices are performed on an asynchronous bus. When a semiconductor is made to be a storage medium, because it can follow operations of the CPU, it becomes significant to make it a synchronous operation.

[0010]

However, as a problem in such a case, there is the issue that the CPU processing bus width and the data access bus width of one chip of flash memory are different. Although this bus width difference can be solved by using memory chips in parallel in cases of DRAMs etc., flash memory has a fixed deleting unit where the minimum deleting unit is 512 bytes, for example. In this case, if multiple chips are utilized in parallel, the capacity of a deletion unit that is deleted at once becomes ($512 \times$ the number of parallel chips) bytes. Thus, the deleting target capacity is so large that data other than the deleting target are also deleted.

[0011]

On the other hand, because personal computer file management currently regards this 512 byte unit as one sector, changing this storage capacity unit for file management (also called the file management unit, below) should be avoided. Therefore, in accessing one sector which is the file management unit, if attempting to utilize four flash memory chips in parallel, the minimum deleting unit becomes larger as a result, and rewriting the sector will require deleting four times the capacity of one sector as described earlier. In order to avoid such a situation, if the CPU has a 32-bit bus and the flash memory has an 8-bit bus for example, one chip needs to be accessed four times sequentially to deal with the 32-bit bus of the CPU.

[0012]

However, when continuously accessing four sectors or more, there is no problem even if four chips are accessed simultaneously in parallel for each four-sector data and 32 bits from the CPU data bus are accessed simultaneously with a one-time access. Namely, if 512 bytes in the first sector are written by 32 bits at a time over 16 times, $128 (16 \times 8)$ bytes are dispersed to each step. Next, 512 bytes in the second sector are written by 32 bits at a time over 16 times. In the same way, 512 bytes in the third sector, and further 512 bytes in the fourth sector are written. At this point of time, all the capacity of the minimum deleting units of four chips is filled with data. Therefore, there is no problem in deleting this four-sector capacity at once when rewriting this file.

[0013]

Thus, simultaneous four-step parallel access is repeated for each four sectors of a file. If there are data left which do not fill the last four sectors, other measures will become necessary.

[0014]

If there are three sectors left, because 32 bits of the CPU cannot be divided by 3, they are separated into two sectors and one sector for handling.

[0015]

If there are two sectors left, or the file is two sectors in size in the first place, 32-bit data from the CPU are divided in two, and two chips are simultaneously accessed by 16 bits each at a time. The 512 bytes in the first sector are written onto two chips by 16 bits at a time over 32 times, where 256 (32×8) bytes are dispersed onto each chip. In the same way, the 512 bytes in the second sector are written onto two chips by 16 bits at a time over 32 times.

[0016]

If there is one sector left, or the file is one sector in size in the first place, 32-bit data from the CPU are divided into four, and one chip is accessed by 8 bits at a time. The 512 bytes in the one sector are stored in one sector of the chip over 64 times.

[0017]

As with the above, changing the method of storing data according to the number of continuously accessed sectors contributes to an access speed increase and solves the deletion problem accompanying flash memory. Namely, in either storing method, because it is guaranteed that only data of the same file are stored inside the minimum deletion area of a chip, the problem accompanying the deletion of flash memory is solved.

[0018]

Here, although the flash memory is operated synchronously in this invention, delays occur according to the access scheme. For this problem, a wait status can be arbitrarily requested to the CPU by using a CPU having a ready signal input. CPUs having a ready signal input are popular today. For example, it is installed in all CPUs having 16 bit processing or higher by Intel Corp., which are the most popular as CPUs mounted in general-use personal computers. Because the CPU halts its processing cycle by negating this ready signal, if reading/writing of access data has not been completed, this signal can be just negated, and when the access is complete, the CPU resumes execution of the processing by asserting it. Because this is a control which becomes possible only when the file storage device is in a synchronous operation with the CPU, inputting synchronized clocks to the both at this time to make them operate synchronously is indispensable. By these controls, even when the data width per single processing is different between the CPU and the flash memory, it can be easily dealt with by letting the CPU wait until the data widths are adjusted.

[0019]

It is usually impossible in file management to separate anything dealt with in a continuous sector once. Namely, a file system can only access in file units. Therefore, when accessing multiple sectors continuously, if multiple chips are accessed simultaneously at the time of writing, using the multiple sectors in parallel, the file data as they were written can be obtained at the time of reading through the same parallel access.

[0020]

As in the above, according this invention, in an information processing device mounted with file storage devices having flash memories as its storage media, synchronous control with the system is performed so that the maximum performance can be drawn in its access performance which is highly superior to magnetic disk devices. In such a case, the CPU is put in a wait status upon necessity by a ready signal input to the system CPU to adjust the timing. Even when the system data bus width and the number of access data bits of one chip are different, it becomes possible to access at as high a speed as possible.

[0021]

FH 008714

The effect is especially great in reading/writing of large capacity files whose access speed is easily recognized by the user, namely in continuous sector accesses. Also, whereas the system data bus width and the number of the access data bits of one chip are different according to the purpose, performance, and age, highly-flexible configuration is made possible to deal with various things. Also, it can be applied to methods to access memory with interleaves to increase the speed.

[0022]

According to the configuration equipped with a chaining information, the access procedure from the system is simplified to support a high-speed access. Then, the file management itself can be simplified, and simplification of control circuit, control program, etc. can be achieved.

[0023]

Also, by being equipped with a data distribution function inside a memory element, reduction of peripheral circuits and data processing speed increase can be achieved.

[0024]

[Embodiments]

Embodiments of the present invention are explained hereafter, with reference to the drawings.

[0025]

First of all, shown in Fig. 1 is a standard personal computer configuration where the flash memory file device of this invention is installed. In the figure, 1 is a CPU which in charge of processing data and programs, and its data bus width is assumed to be 32 bits. No. 2 is a clock generator which generates a synchronous clock for the whole system. No. 3 is a system internal bus containing a data bus, an address bus, memory commands, I/O commands, etc. No. 4 is a file control circuit which performs file management and memory control of the flash memory file device of this invention. No. 5 is a flash memory array which becomes a storage medium of the flash memory file device, where the number of access data bits per chip is 8. No. 6 is a main memory control circuit which manages/controls the system main memory, and 7 is DRAM which is the main memory. No. 8 is a peripheral I/O bus control circuit, where a display control circuit 9 and a display 10 are connected as one of peripheral I/O devices. Listed as other peripheral I/O devices are a communication device 11, an external large capacity storage device 12, etc.

[0026]

Usually, the peripheral I/O bus control circuit contains another clock generation circuit, and these peripheral I/O devices operate based on that clock cycle. However, there can be one which is directly connected to the internal system bus 3 for speed enhancement and operates in synchronization with the CPU 1. No. 13 is a clock signal supplied to each circuit containing the CPU 1 for synchronizing circuits connected to the internal system bus 3. Note that it does not need to be entirely the same with the clock supplied to the CPU 1 but can be a divided one depending on the circuit as far as it is synchronized.

[0027]

No. 14 is a ready control signal input to the CPU 1, where the status indication signals output from individual circuits are general-controlled in a ready control circuit 15 and input to the CPU 1. No. 16 is a control circuit of input devices for the user of this system to instruct the desired processing, and 17 is an input device. In the figure, the input device is assumed to be a keyboard, and its control circuit 16 is assumed to be a keyboard controller (KBDC).

[0028]

Next, actions of the system in Fig. 1 are explained. During normal operation, the CPU 1 computes/processes programs and data stored in the main memory 7 to execute the process that the user instructed through the input device 17, and displays the result on the display 10. Also, if necessary, it starts up the communication device 11 and stores a large capacity of data to the large capacity external storage device 12. Then, in retrieving or storing files, it operates the flash memory file device consisting of the file control circuit 4 and the flash memory 5. When booting up the system also, the system program is loaded from here. During these operations, synchronous operations are performed by the clock 13 generated by the clock generator 2, and if there occurs a need for a certain circuit to request a stand-by to the CPU 1, that circuit request a CPU stand-by to the ready control circuit 15, and it is conveyed to the CPU 1 by negating the ready signal 14. The CPU 1 will continue to stand by until the ready signal 14 is asserted again. At this time, the file control circuit 4 can perform a control of increasing/decreasing the CPU stand-by time by the number of files requested to the CPU 1. This mechanism is explained with reference to Fig. 2.

[0029]

Figure 2 is a drawing to explain the internal configuration of the flash memory file device. In the figure, 3, 4, 5, and 13 are the same as those in Fig. 1, and the following are internal components of the file control circuit 4. No. 21 is a register group which performs interfacing with the system bus 3, no. 22 is a status register for reporting the status of the file control circuit 4 to the CPU 1, 23 is a register for setting the starting sector number to access, 24 is a register for setting the ending sector number to access, 25 is a command register which instructs the processing requested by the CPU 1 as a command code, and 26 is a data register which becomes a window of data give and take with the system bus. 27 is a controller which takes a general control of internal controls of the file control circuit 4, which is ideally a programmable, intelligent LSI such as a one-chip microcomputer. No. 28 is a control circuit which performs control of the flash memory 5 which is a storage medium, 29 is a data control circuit which performs control of data to read from or write to the flash memory. No. 30 is a DMA control circuit for performing memory access at high speed, and the system clock signal 13 is input to this circuit. No. 31 is a status indication signal output to the ready control circuit 15, no. 32 is a local bus inside this flash memory file device, and 33 is a control signal and address for accessing the flash memory.

[0030]

Next, actions of the flash memory file device in Fig. 2 are explained. When a necessity to access with the flash memory file device arises, the CPU 1 performs the access via the system bus 3. In order to do so, first the content of the status register 22 is read out to check whether it is in an accessible status or not. Then, the sector to access next is set to the starting sector register 23 and ending sector register 24. Then, it writes the command code (read or write) of the requested access to the command register 25. Then, the status register 22 is read again, and if the access is possible, data are written to or read from the data register 26.

[0031]

At this time, the controller 27 manages these interface registers 21 to respond to requests from the CPU 1. Namely, it reads the starting sector register 23, ending sector register 24, and command register 25, grasps the access content of the flash memory 5, write a code indicating the current status to the status register 22, and reports to the CPU 1.

[0032]

Because it is expected that the controller 27 is too slow in its operation speed to perform directly flash memory access for responding to the access requests from the CPU 1, the configuration is made so that the DMA control circuit 30 performs flash memory access at high speed and performs give and take of data with the system bus 3. The controller 27 will perform setting the access content, starting up the DMA, etc. to the DMA control circuit 30 and the memory control circuit 28 for that purpose.

[0033]

The DMA control circuit 30 performs address generation and timing generation for performing the DMA, and the memory control circuit 28 generates access signals follow that timing. The flash memory 5 performs give and take

of data with the data control circuit by these input signals. The data control circuit 29 performs data generation according to the number of sectors to access.

[0034]

For example, in the case of write access of one sector, data sent by one-time access from the system bus 3 are adjusted to the number of writing bits of one chip of flash memory. In this embodiment, because the data bus width of the system bus 3 is set to 32 bits, 32 bits of data are obtained through a transfer by one access. Then, because the data width of one chip of flash memory is set to 8 bits, the sent data are divided into four to write to the flash memory 5. Therefore, the processing of dividing 32-bit data into four 8-bit data is performed by the data control circuit 29 using a latch circuit. Also, in the case of read access of one sector on the contrary, the flash memory 5 is read accessed four times to prepare 32-bit data for one bus transfer. The wait time of the system bus occurring at this time is generated by a stand-by request to the CPU 1 by a status indication signal 31 generated by the data control circuit 29.

[0035]

On the other hand, in the case of access to multiple sectors, the data control circuit 29 adjusts the data to latch for increasing the access speed and execute it normally. For example, if it is a continuous read access of four sectors, because 32-bit data are completed by simultaneously reading four chips of 8-bit access flash memory 5, the wait time of the system bus 3 can be greatly reduced. Note that four-chip simultaneous write access needs to be performed at the time of writing to the flash memory 5. Otherwise, the order of data will come to differ and the file data will become abnormal. Note that because it is common to perform management with file units in systems dealing with file storage devices, it is usual to read-access the same number of sectors with the number of sectors when they were written, and if the access schemes according to the number of sectors are made entirely the same between writing and reading, there will be no need to record the information. Namely, if it is an access scheme of continuous writing of five sectors, the first four sectors are simultaneously written to four chips in parallel, and the remaining one sector is divided in four to one chip, adopted in both reading and writing for example, and normal file data can always be accessed. Here, to make sure, information on the data storing scheme may be recorded in data in sector units stored in the flash memory 5. As its record location, if there exists a redundant area other than the data storing area of the flash memory, storing there is appropriate, and if there is no redundant area, a separate storing area is installed for recording.

[0036]

A continuous access of six sectors is dealt with a parallel access of four sectors and a parallel access of two sectors.

[0037]

Next, these access signals and data control method are explained in even more detail with reference to Fig. 3.

[0038]

Shown in Fig. 3 is a configuration where the bit width of the system bus is set to 32 bits, and that of the flash memory to 8 bits. In the figure, numbers that already appeared before are identical to those explained so far. Newly, 41 is a counter for generating addresses of the DMA control circuit 30, where the system clock 13 and I/O access (command) signals or memory access (command) signals of the system bus 3 are input and counting up is performed in synchronization with this. No. 42 is a start-up register for DMA control where the local bus 32 of the controller 27 is connected, and a desired DMA transfer can be started by writing a code to this register. No. 43 is a sector register which is connected to the local bus 32, too, and DMA transfer of an arbitrary sector number can be performed by writing the sector number to access. In the actual operation, written value of this sector number is input to the memory control circuit 28 which is used for generating the upper-order address of the flash memory and chip select signal. No. 44 is a timing circuit which generates a timing signal for synchronizing in each control circuit at DMA transfer. No. 45 is a memory address generated by the memory control circuit 28 based on values of the counter 41 and sector register 43. No. 46 is a memory control signal generated by generation of the memory address 45. 51, 52, 53, and 54 are data latches of one byte (8 bits) each, constituting data latches of four bytes (32

bits) for data width conversion between 32-bit data and 8-bit data. When data of the system bus 3 are denoted as D0~D31, the latch 51 latches D0~D7, the latch 52 D8~D15, the latch 53 D16~D23, and the latch 54 D24~D31, respectively. 55 is a latch signal generation circuit for these data latches. 56 is a data control setting register which is connected with the local bus 32 and sets data width and data order. In this embodiment, generation of a latch signal and its timing are instructed to the latch signal generation circuit 55 by setting one of "1", "2", and "4" to this data control setting register 56 as the number of continuous access sectors at read access. Then, latch signals input to data latches 51, 52, 53, and 54 are 57, 58, 59, and 60, respectively. For example, if "1" is set, latch signals 57, 58, 59, and 60 are output sequentially one by one, and data from one chip of flash memory are accessed four times to generate 32-bit data which are output to the system bus 3. Also, if "2" is set, latch signals 57 and 58 are output simultaneously and afterwards latch signals 59 and 60 are output simultaneously, which are performed alternately, and data from two chips of flash memory are accessed twice each to generate 32-bit data which are output to the system bus 3. Also, if "4" is set, latch signals 57, 58, 59, and 60 are output simultaneously, and data from four chips of flash memory are accessed only once to generate 32-bit data which are output to the system bus 3.

[0039]

On the other hand, during a write access, latch signals are always sent to data latches 51, 52, 53, and 54 simultaneously to latch 32-bit data from the system at once. No. 61 is a data distribution circuit which distributes data stored in data latches 51, 52, 53, and 54 or data from the memory 5. No. 62 is a 32-bit data bus connecting the flash memory 5 and the data distribution circuit 61. No. 63 is a read/write signal for determining the data direction of the data distribution circuit 61, which should better receive supply from the command register 25 of the interface register 21. The data distribution circuit 61 is made as a bidirectional buffer, where one side is connected to data latches 51, 52, 53, and 54, and the other side to the flash memory 5. On the flash memory 5 side, the input is in 32 bits, and all the chips of the flash memory 5 are divided into four groups, which are divided into separate bit groups (Bits 0~7), (Bits 8~15), (Bits 16~23), and (Bits 24~31), and input as 32 bits in total. Then, the direction is determined by whether it is read or write, and partitioning of each data are determined by the setting content of the data control setting register 56.

[0040]

Specific explanations are given on this distribution of data using Fig. 12, Fig. 13, and Fig. 14. These figures show data distribution examples of the data distribution circuit 61 for different values set to the data control setting register 56. Figure 12 shows data distribution when the number of continuous access sectors is 1, Fig. 13 when the number of continuous access sectors is 2, and Fig. 14 when the number of continuous access sectors is 4, respectively, and there are four kinds of data distributions in each case depending on which of the four memory groups to access. Also, read and write are distinguished. When the number of continuous access sectors is 1, four system cycles make one access, when the number of continuous access sectors is 2, two system cycles make one access, and when the number of continuous access sectors is 4, one system cycle makes one access.

[0041]

When the number of continuous access sectors is 1, classification into four kinds is made depending on where in the divided memory to access, and when the number of continuous access sectors is 2 or 4, classification into four kinds again is made depending on which memory group is set as the starting point. Namely, the starting point can be adequately determined depending on how the memory is used, and by doing so the bias of used memory groups can be prevented. For example, although the data distribution method can be simplified if the memory group 1 is always set as the starting point, it is expected that the ratio of usage of the memory group 1 becomes high and there occur biases in both the amount of use and the frequency of use. If a bias occurs, one memory group eventually becomes unavailable, making it impossible to perform high-speed write access when the number of continuous sectors is 4. Therefore, it is made possible to set the starting point in any memory group.

[0042]

Note that although in reading when the number of continuous sectors is 1 or 2, data connection lines in the distribution circuit are made separate for each cycle in the figure, because the latch signal is not output to other than the target latch, the connection line may be shared instead of separating for each cycle. Namely, in reading of A

where the number of continuous sectors is 1 (Fig. 12), whereas the distribution is performed in a sequential cycle with latch 1 – memory group 1, latch 2 – memory group 1, latch 3 – memory group 1, and latch 4 – memory group 1, connection lines to all the latches may be connected to 1 of the memory group in all cycles. This is because even if data lines are connected, there is no influence if no latch signal is output.

[0043]

More specifically, referring to Fig. 12 in the write access of one sector for example, one byte connected one of the access target flash memory groups among the 32-bit data bus 62 is written to the flash memory with the data latch 51 at the first time, the data latch 52 at the second time, the data latch 53 at the third time, and the data latch 54 at the fourth time, which is repeated from 51 sequentially. In the read access, one cycle of data are distributed to each latch by the latch signal.

[0044]

In the write access of two sectors, namely, if it is a write with a data width of 2 bytes, from Fig. 13, as 2-byte data connected to the corresponding two memory groups among the 32-bit data bus 62, data are received from the data latches 51 and 52 at the first access, 53 and 54 at the second access, and distributed alternately afterwards. On the other hand, in the read access the order becomes reversed, and data are distributed from the corresponding two memory groups to the data latches 51 and 52 in the first access, 53 and 54 in the second access, and this is repeated.

[0045]

In the write access of four sectors, namely, if it is a write with a data width of 4 bytes, the buffer direction becomes the one from the data latch to the flash memory 5, the 32-bit data bus 62 is connected to the data latches 51, 52, 53, and 54 sequentially from its top memory group, and 32-bit access is completed by one-cycle access. In reading, the direction becomes the opposite.

[0046]

In order to perform the actions, the controller 27 needs to set appropriate values to the registers explained so far before starting up the DMA control circuit 30.

[0047]

When multiple memory groups are accessed simultaneously in order to increase the speed as described earlier, addresses given to the individual memory groups are investigated below referring to Fig. 5.

[0048]

In Fig. 5, 81~84 indicates memory groups 1~4, respectively. The same figure (1) shows a state where a certain amount of file writing has been performed, and the same figure (2) shows a state where a certain file has been updated and its file size has increased. The data code (m-n) shown in the figure indicates a file number m and a sector number n in each file. For example, (3-2) indicates the second sector of the file number 3. Note that this figure shows area (having a capacity of one sector or more) of which memory group is used how as a whole, and in actuality, at a continuous access of two sectors or more, instead of content of only a single sector being stored in a one-sector capacity area of each memory group, contents of multiple sectors are stored dispersed. This point will be described in detail later in the explanation of file management.

[0049]

In Fig. 5 (1), in the access of file number 1 (all sectors coded as 1-n), in a simultaneous access from 1-1 to 1-4, because these are arranged in the same line, same address can be given to these memory groups. However, in file number 2, if 2-1 to 2-4 are used for a simultaneous access, the memory group 4 needs to be given a different address from other memory groups. This means that the same number of address buses with the number of memories need to be installed. Or, there needs to be some way to give different addresses to the memory groups. In order to avoid

this, an access scheme needs to be taken in accesses of file number 2 that 2-1 is accessed alone, 2-2 and 2-3 are simultaneously accessed, and 2-4 is accessed alone again. However, this will generate a waste in increasing the speed. Then, an example of configuration for giving different addresses is explained using Fig. 9.

[0050]

Figure 9 is an example of configuring an address generation circuit inside the memory control circuit 28 when the number of memory groups is set to 4. In the figure, 201~204 are four memory groups constituting the flash memory 5, where 201 is a first memory group a, 202 is a second memory group b, 203 is a third memory group c, and 204 is a fourth memory group d. 205 is a latch circuit b of the high-order address given to the memory group b, 206 is a latch circuit c giving to the memory group c, and 207 is a latch circuit d giving to the memory group d. 208 is the address bus of the memory control circuit 28 (corresponding to 45 in Fig. 3), and 209 is the low-order address of the address bus 208. The number of its address bits is set to the number of address bits which corresponds to that for accessing the storage capacity of the data deletion minimum unit or the minimum unit for file management of the flash memory chip. For example, if the data deletion minimum unit is 512 bytes, it becomes 9 bits. Only when the data deletion minimum unit of the flash memory chip is smaller than the minimum unit for file management, it is effective to make it the number of address bits which corresponds to the minimum unit for file management. 210 is the high-order address of the address bus 208 excluding the low-order address 209. Note that even higher-order address which is unnecessary in accessing the memory groups is removed. No. 211 is a high-order address stored in the address latch b for accessing the memory group b, 212 is a high-order address stored in the address latch c for accessing the memory group c, and 213 is a high-order address stored in the address latch d for accessing the memory group d. No. 214 is a memory control signal for the memory group a201, 215 is a memory control signal for the memory group b202, 216 is a memory control signal for the memory group c203, and 217 is a memory control signal for the memory group d204.

[0051]

In the configuration of Fig. 9, when accessing the memory group a to the memory group d simultaneously, high-order addresses to access these memory groups are written in advance to the address latches 205~207. Then, when performing the access, because the low-order address 210 is common to all the memory groups, the address bus supplies the address for accessing only the memory group a, and memory groups other than the memory group a are accessed through the high-order addresses from the corresponding address latch circuit. Note that access control is entrusted to the memory control signals 214~217, and if access is only to the memory group a and the memory group b for example, it is arranged so that only the memory control signals 214 and 215 become active. Also, in accessing only the memory group d for example, unless the memory control signal 217 is set active, the memory group a may be given any address without a problem. By this means, it becomes possible to give different addresses to individual memory groups, only if the memory groups are dispersed into 4, data of the same file stored in locations at physically different addresses can be accessed simultaneously, enhancing the contribution of speed increase. In the configuration of Fig. 9, the address latch of the memory group a can be omitted. Of course, if it is effective to install an address latch also for the memory group a, it may be done so.

[0052]

According to the above embodiment, the CPU 1 can perform read/write access of a desired sector at high speed by performing simple specifications to relatively a small number of registers. Also, the file controller is made as a one-chip microcomputer, where fine control can be instructed by software, and data transfer can be performed at high speed by loading a DMA transfer control circuit even if the one-chip microcomputer can only perform slower operations than the system. However, if the operation speed of the one-chip microcomputer is high enough to perform high-speed operations for the system, the DMA control circuit is not necessary, and a configuration where all data transfers are performed by the one-chip microcomputer can also be considered.

[0053]

Also, although an explanation has been given in this embodiment assuming the data buses are 32 bits for the system and 8 bits for the flash memory, also for the CPU 1 of 16-bit operation, CPUs of 64-bit operation, and a flash memory of 16-bit I/O, adjustment to other data widths can be easily performed by changing the number of data

latches, configuration of the data control setting register, and control programs of controllers. The concrete example is shown in Fig. 15. Shown in Fig. 15 are concrete numerical values of hardware configuration in a combination of individual bit widths. The horizontal configuration of the figure is the number of data bits of the flash memory, where 4 bit, 8 bit, 16 bit, and 32 bit are listed as examples. The vertical configuration shows the system data width, where 8 bit for simple information equipment, 16 bit and 32 bit which are the mainstream of the present personal computers, and 4 bit and 128 bit for the future personal computers and high-performance computers are listed as examples. Also, the data distribution circuit listed as a hardware configuration is, if explained using the data distribution circuit 61 of the embodiment, shown in the number of bits of one circle \times the number of circles in the figure. Connection of each circle can be shown in the same idea with Fig. 12 ~ Fig. 14. While the number of latches is increase or decrease of the number of latches 51~54 in the embodiment, if the number of memory data bits is larger than the system data width, the latch location will be configured in the data distribution circuit memory side. The number of memory groups indicates the number of divisions of the memory which is divided in four in the embodiment.

[0054]

Note that the number of distribution of the data distribution circuit, the number of latches, and the number of memory groups are lower limit numbers including this embodiment, and if there is some room in the circuit scale, the number of terminals, etc., increasing them will make a more effective system configuration. That is because increasing the number of memory groups gives more room in selecting memory for storing data, which can prevent biases of memory usage and frequency of use. Also, because the number of memories to access in parallel can be increased, speed increase can be further progressed. Especially, when the system data width and the number of memory data bits are equal to each other or the number of memory data bits is larger, although there is no effect of adjusting the data bus widths at all as described in the embodiment, multiple memory chips can be simultaneously accessed and speed increase by the so-called interleave method can be realized. In this case, at the same time with increasing the number of memory groups, the number of memory data distributions in the data distribution circuit and the number of latches need to be increased.

[0055]

Note that although explained in this embodiment is that the flash memory can follow data transfer from the system, to do so it is preferable that a write buffer be installed inside the flash memory. If it is a flash memory which can perform writing at high speed in the future, this kind of write buffer will become unnecessary. If no write buffer is contained in the flash memory, a write buffer can be installed between the flash memory and the data control circuit. Then, at each write access, writing is performed not directly to the flash memory but first to the write buffer, and after data transfer from the system is complete, writing is performed from the write buffer to the flash memory. The configuration diagram of an embodiment of this case is shown in Fig. 4.

[0056]

In Fig. 4, numbers which already appeared before are the same with those explained so far. As new ones, 71 is a data selector which switches connection with the data distribution circuit 61 depending on whether the access is read or write, 72 is a write buffer of 32-bit width which temporarily stores write data, and 73 is a read/write signal for switching the data selector 71, where a part of data of the command register 25 is better to be input. If the access is write, the data distribution circuit 61 is connected with the write buffer, write data are stored in the write buffer, and the controller performs writing to the flash after data transfer from the system. If it is read, the data distribution circuit 61 is connected directly to the flash memory, and the same operation with the read access in Fig. 3 describe above is executed. By making it the configuration, even using a flash memory which does not contain a write buffer and has a slow writing, high-speed accesses become possible, seen from the system.

[0057]

Next, an embodiment of file management is explained with Fig. 5 as an example. Although embodiments of hardware have been explained so far, how to store actually to the flash memory is described below. This operation is basically performed by the system CPU, system program, and the controller of the flash memory files, where the central issue is operation by software. In the explanation using Fig. 5, instead of the content of the software itself,

FH 008721

shown is how unit capacity area (for one sector) of each memory group of the flash memory as the whole is assigned to a sector of a file as a result of the operation. For each sector of a file, unit capacity area is secured in sequential memory groups. For continuous sectors of four or more, one unit capacity area of sequential memory groups 1~4 are assigned to the four sectors (1-1~1-4 in the figure for example). Note that data in each sector are actually stored dispersed in four unit capacity areas. To two sectors (1-5~1-6 in the figure for example), unit capacity areas of two memory groups are assigned. In this case also, data in each sector are stored dispersed in two unit capacity areas. Assigned to the last sector 1-7 of file 1 is a unit capacity area of a single memory group.

[0058]

Here, an even more detailed explanation is given on dispersed storing of two sectors and four sectors using Fig. 10 and Fig. 11. Shown in Fig. 10 is an example of 4-byte simultaneous write, and Fig. 11 is an example of 2-byte simultaneous write. In the same ways as the above, it is assumed that the host system bus is 32 bits, and that the memory access width of one memory group is 8 bits.

[0059]

Shown in Fig. 10 is an example of 4-sector simultaneous write to the memory groups a~d. In the figure, indicated with 211 is data from the system bus, which are sent with 32 bits (4 bytes) as a unit. Indicated with 222~225 are sequential numbers of 8-bit (1-byte) unit data, 222 is the first byte data, 223 is the second byte data, 224 is the third byte data, and 225 is the fourth byte data. Following them, data up to the 2048th byte are continuously sent as those for 4 sectors. Actually, because of the 32-bit bus, from the first byte to the fourth byte are simultaneously received, and the following are the same. Nos. 51~54 are data latches for temporary storage of data shown in Fig. 3.

[0060]

Because of being simultaneously written 4 bytes at a time, after storing 32 bits, namely 4 bytes, simultaneously to the data latches 51~54, each byte is dispersed to the memory groups a~d and is written one byte each. For the next 32-bit data, each byte is written to the following parts of the memory groups a~d. In this way, when 512 bytes corresponding to 1 sector are written, subsequently data for the second sector are written by 4 bytes at a time to the following parts of the memory groups a~d in the same way. Although Fig. 5 is shown for convenience as if each sector is assigned to a specific memory group, as in the above explanation, actually data in each sector are not stored only in a specific memory group but written dispersed to the memory groups a~d. Note that it is true that data of the whole 4 sectors are stored in an area of 4-sector capacity in the memory groups a~d.

[0061]

Shown in Fig. 11 is an example of 2-byte simultaneous write to the memory group a and the memory group b.

[0062]

Because of being a simultaneously written 2 bytes at a time, after storing 32 bits, namely 4 bytes, simultaneously to the data latches 51~54, these are divided in two by 2 bytes each, the first half 2 bytes are first written to the memory group a and the memory group b by 1 byte each, next the latter half 2 bytes are written to the memory group a and the memory group b by 1 byte each. In this way, when 512 bytes corresponding to 1 sector are written, subsequently data for the second sector are written from the first byte to the following parts of the memory groups a and b in the same way. In this case also, the first sector is not stored only to a specific memory group but written dispersed to the memory groups a and b, and in the same way, the second sector is written dispersed to the memory groups a and b following the first sector.

[0063]

Although writing to a single sector is not shown, individual byte data of the data latches 51~54 are simultaneously stored to a certain memory group sequentially in this case. Only in this case, data of one sector are stored to a specific memory group without being dispersed.

FH 008722

[0064]

As can be seen from the above, the important thing is that there is no data mixed from different files in the area of the minimum deletion unit (512 bytes corresponding to one-sector capacity here) of a flash memory element as a result of the above operations. By this guarantee, it becomes possible to achieve high-speed file data storing by writing multiple bytes simultaneously in parallel as effectively utilizing the storage area of memory when rewriting a file without deleting data of other files.

[0065]

As stated above, Fig. 5 (1) shows a state where a certain degree of file writing has been performed, Fig. 5 (2) a state where a certain file has been updated and the file size has increased afterwards. In Fig. 5 (1), storing is performed so that different memories are assigned in the order of file numbers to be stored and the order of sector numbers. In principle, they are packed to leave no space so that there will be no waste in terms of management.

[0066]

As shown in Fig. 5 (1), if file sectors are stored in memory groups by packing sequentially and continuously so that no vacant space occurs, when a file size has increased by updating the file, there occurs cases as shown in Fig. 5 (2) where continuous storing in terms of physical storage location is impossible. Even in such a case, storing is performed so that the order of memory groups becomes continuous. Namely, if the file number 4 is taken as an example, while five sectors from 4-1 to 4-5 are stored in (1), if two sectors are added in (2), because 4-1 is started from the memory group 3, areas in the memory groups 2 and 3 are secured for the increased portion of 4-6 and 4-7. When writing a file whose number of sectors is increased, 4-5 and 4-6 are treated as 2-sector continuous sectors at this time, and 4-7 is treated as a single sector. Namely, added sectors also receives a treatment as continuous sectors together with already-stored sectors. In order to perform smoothly these storing and reading out in file management, it is believed to be more appropriate as a hardware configuration that specification of access sectors from the system should be the starting sector and the number of sectors and grasping the physical storage locations should be performed inside the file system. Namely, whereas specifying access files from the system should be simplified as much as possible for the speed increase, if physical locations are scattered, specifying the physical locations becomes complicated. Therefore, the ending sector 24 shown in Fig. 2 should be removed in this sense. Then, by storing information indicating file chaining in an redundant area of the memory to store information other than data if it exists, or in another storage means if it does not exist, because if the file starting number is specified, all the physical locations of sector numbers following it become clear by chaining, a continuous access becomes possible.

[0067]

Shown in Fig. 6 is an example of chaining information. In the figure, 85 is file data stored in memory, which are data of file number 4 and sector number 5. 85 is the data of file number 4 and sector number 6 that follow them. And 87 is data of file number 4 and sector number 7 that further follow them. Note that although these data are arranged in the order of the sector numbers, they may be mixed in the actual storage state. Namely, data may be stored scrambled by accessing multiple chips simultaneously through a continuous sector access. However, because numbering of data storage by byte is necessary even in such a case, sector numbers are very important. 88 is a physical address indicating the physical location on memory where the file data 85 are stored, where the number "3" on the left side is the memory group number, the number "5" on the right side is the address in that memory group. 89 and 90 are physical addresses of the file data 86 and the file data 87 in the same way. 91 is chaining information indicating the physical address where the next sector of the file data 85 are stored, and because the file data 85 are data of file number 4 and sector number 5, the chaining information 91 indicates the physical address where data of file number 4 and sector number 6 are stored. The number on the left side is the memory group number, and the one on the right side is the address in the memory group. In the same way, chaining information 92 indicates the physical address where data of file number 4 and sector number 7 are stored, and chaining information 93 indicates a case where the next sector does not exist. Namely, it can be told that the file of file number 4 ends with 7 sectors. If this chaining information is attached with file data, the system CPU does not have to grasp the physical location of file storage but can take a scheme of simply accessing the file itself. Then, by the controller of file control performing physical accessing of memory as referring to the chaining information, even with a file whose physical storage locations are not continuous but scattered, a continuous access becomes possible. In that case, a

configuration to input different addresses for simultaneous access of different memory groups is necessary. Note that as stated earlier in Fig. 9, low-order address for the one-sector access can be shared, necessity is only for the high-order address for more than that. According to embodiments using chaining information above, access specification from the system can be simplified to enable the execution with smaller amount of information.

[0068]

Next, an explanation is given on an embodiment of embedding various kinds of functions in a memory element itself.

[0069]

Figure 7 is a configuration diagram of a memory element where multiple memory chips are contained in a package and an additional circuit is installed. In the figure, 101 is a memory package, 102~105 are memory chips having the same function, 106~109 are input/output data terminals of the memory package 101, each of which can perform data input/output of one memory chip. For example, if the memory chip 102 has 8-bit data input/output, 106~109 come to have $8 \text{ bits} \times 4 = 32 \text{ bits}$ of data input/output terminals in total. No. 110 is a signal input terminal which specifies the path to connect these memory chips with input/output terminals, 111 is a specification signal, and 112 is a connection path setting circuit. A user of this memory 101 executes an access after setting 109 with the signal input terminal 110 from the data input/output terminal 106. It is considered as an example that this setting is selected from connection paths such as those in Fig. 12, Fig. 13, and Fig. 14 stated earlier. In the memory 101, set data connection is sent as the specification signal 111 to the connection path setting circuit 112, the connection setting circuit 112 connects the memory chips 102~105 with the data input/output terminals 106~109 according to the setting, and realizes connection of data in the memory chip with the desired data bus.

[0070]

Figure 8 is a configuration example of a memory element which performs setting of data connection from a user by a command input. In the figure, 113 is a command control circuit which creates the specification signal 111 from a register of the command set value and its set value. 114 is a data line where the user set the command set value using a part of the data input/output terminal. Others are the same as those with the same numbers in Fig. 7. A user selects a data connection path with Fig. 12, Fig. 13, and Fig. 14 as an example, sets as a command code from a part of data bus 114 to the command control circuit 113, and sends the specification signal 111 which corresponds to the connection path setting circuit 112 from the command code set in the command control circuit 113. Below is the same as the explanation of Fig. 7. According to this embodiment by Fig. 7 and Fig. 8, another embodiment explained so far can be realized as a memory element, and there is an efficacy that reduction of peripheral circuits can be achieved. Note that although in this embodiment multiple chips and control circuits are to be loaded in a memory package, by uniting these into one chip, miniaturization and speed increase can be promoted.

[0071]

[Efficacy of the Invention]

According to the present invention, it becomes possible in a flash memory device to achieve high-speed file data storing by writing multiple bytes simultaneously in parallel as effectively utilizing the storage area of the memory when rewriting a file without deleting the data of other files.

[Brief Explanation of the Drawings]

[Fig. 1] A block diagram showing a system configuration which realizes this invention.

[Fig. 2] A block diagram showing the configuration of an embodiment of the flash memory file device.

[Fig. 3] A block diagram of the main section for explaining actions of the embodiment.

[Fig. 4] An explanatory drawing showing data control when a slow-writing flash memory chip is used in the embodiment.

[Fig. 5] An explanatory drawing of an execution example of storing files to the memory in the embodiment.

[Fig. 6] An explanatory drawing of an execution example of storing chaining information in another embodiment of this invention.

[Fig. 7] An explanatory drawing of a memory configuration example where the data distribution function by the signal terminal input specification is installed in the memory element.

[Fig. 8] An explanatory drawing of a memory configuration example where the data distribution function by the command setting input specification is installed in the memory element.

[Fig. 9] An explanatory drawing of address connection of the memory groups in the embodiment.

[Fig. 10] An explanatory drawing of actions at 4-byte simultaneous write in the embodiment.

[Fig. 11] An explanatory drawing of actions at 2-byte simultaneous write in the embodiment. (system bus 32 bit)

[Fig. 12] An explanatory drawing of data distribution at memory 1-chip simultaneous access in the embodiment.

[Fig. 13] An explanatory drawing of data distribution at memory 2-chip simultaneous access in the embodiment.

[Fig. 14] An explanatory drawing of data distribution at memory 4-chip simultaneous access in the embodiment.

[Fig. 15] An explanatory drawing of hardware configurations in various configurations of the system bus and the memory data bus.

[Explanations of the Codes]

1: CPU, 2: Clock generation circuit, 3: System bus, 4: File control circuit, 5: Flash memory, 6: Main memory control circuit, 7: Main memory, 8: External I/O control circuit, 9: Display control circuit, 11: Communication circuit, 12: Large capacity external storage device, 13: System clock, 15: Ready control circuit, 16: KBDC, 21: I/F register, 22: Status register, 23: Starting sector register, 24: Ending sector register, 25: Command register, 26: Data register, 27: Controller, 28: Memory control circuit, 29: Data control circuit, 30: DMA control circuit, 31: Status indication signal, 32: Local bus, 41: Counter, 42: Startup register, 43: Sector register, 44: Timing control circuit, 45: Address, 46: Control, 51: Latch circuit, 52: Latch circuit, 53: Latch circuit, 54: Latch circuit, 55: Latch circuit generation circuit, 56: Data control setting register, 61: Data distribution circuit, 63: Read/write signal, 71: Data switching circuit, 72: Write buffer, 73: R/W signal, 85: Data (4-5), 86: Data (4-6), 87: Data (4-7), 91: Chaining information, 101: Memory package, 110: Specification signal input terminal, 112: Connection path setting circuit, 113: Command control circuit, 201: Memory group a, 202: Memory group b, 203: Memory group c, 204: Memory group d, 205: Address latch b, 206: Address latch c, 207: Address latch d, 208: Address bus, 209: Low-order address, 210: High-order address, 214~217: Memory control signals.

(Continuation of the front page)

(72) Inventor: Jun Kitahara
c/o Microelectronics Equipment Development Laboratory
Hitachi, Ltd.
292, Yoshida-cho, Totsuka-ku, Yokohama City, Kanagawa Pref.

(72) Inventor: Yasuhiro Hida
c/o Microelectronics Equipment Development Laboratory
Hitachi, Ltd.
292, Yoshida-cho, Totsuka-ku, Yokohama City, Kanagawa Pref.

(72) Inventor: Kazunori Furusawa
c/o Semiconductor Design Development Center
Hitachi, Ltd.
20-1, Jōhsui-Honcho 5-chome, Kodaira City, Tokyo

FH 008726

Japanese/English Technical and Patent Translations

**ROGER P. LEWIS
3206 LOCK GATE CT.
HERNDON, VA. 22071
TEL. (703) 437-1878
FAX. (703) 995-0669**

September 16, 2002

TO: Brian Fish c/o
Finnegan Henderson

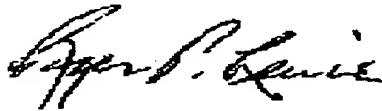
FROM: Roger Lewis

Transmitted herewith is the translation of JP Hei 6-266596, the word count of which is 11,565 words.

The invoice will be sent separately.

Thank you for your consideration, which is appreciated.

SINCERELY,

A handwritten signature in cursive script, appearing to read "Roger P. Lewis".

ROGER P. LEWIS

(19) Japan Patent Office (JP) (12) Public Patent Announcement (A) (11) Patent Application Publication
Pat App Hei 6-266596
 (43) Publication: September 22, 1994

(51) Int. Cl. ⁵	ID Code	Internal Cl. No.	FI	Technology indication section
G 06 F 12/00	520 P	8944-5B		
	550	9366-5B		
Examination request: Not requested			Number of patent Claims: 11	OL (Altogether 21 pages)

(21) Application No.: Pat App Hei 5-51041 292, Yoshida-cho, Totsuka-ku, Yokohama City, Kanagawa Pref.

(22) Application Date: March 11, 1993

(71) Applicant: 000005108
 Hitachi, Ltd.
 6, Surugadai 4-chome, Kanda,
 Chiyoda-ku, Tokyo

(72) Inventor: Kenichi Kaki
 c/o Microelectronics Equipment
 Development Laboratory
 Hitachi, Ltd.
 292, Yoshida-cho, Totsuka-ku,
 Yokohama City, Kanagawa Pref.

(72) Inventor: Kunihiro Katayama
 c/o Microelectronics Equipment
 Development Laboratory
 Hitachi, Ltd.

(74) Agent: Kazuko Tomita, Patent attorney

(Continued to the last page)

(54) [Title of the Invention] Flash memory file storage device and information processor.

(57) [Abstract] (Modifications exist.)

[Construction]

In a file storage device using flash memory elements whose minimum deleting unit is larger than X bits and access width is x bits ($x = X/p$: p is an integer equal to or greater than 2) through an X-bit data bus, it is equipped with a flash memory device 5 consisting of p sets of flash memory element groups accessible simultaneously, a dividing means 51~54 which divide data on the data bus into at least p parts, a data distributing means 61 which has at least a first function to make p pieces of x-bit data correspond to one set of the flash memory element groups and a second function to make each of the p pieces of x-bit data correspond to another set of the flash memory element groups, and a control means 56 which controls the data distribution means 61 so that the first and second functions are switched according to the number of storage capacity units for file management of the access target file.

[Efficacy]

It becomes possible to read multiple bytes simultaneously in parallel while effectively utilizing the memory area.

[Scope of Patent Claims]

[Claim 1]

In a flash memory file storage device which is a file storage device storing data through an X-bit data bus and which uses flash memory elements whose minimum deleting unit is larger than X bits and the access width is x bits ($x = X/p$; p is an integer equal to or greater than 2), a flash memory file storage device characterized by being equipped with a flash memory device consisting of p sets of flash memory element groups accessible simultaneously, a dividing means which divide data on the X-bit data bus into at least p parts, a data distributing means which has at least a first function to make p pieces of x-bit data obtained by the division correspond to one set of the flash memory element groups and a second function to make each of the p pieces of x-bit data correspond to another set of the flash memory element groups, and a control means which controls the data distribution means so that the first and second functions are switched according to the number of storage capacity units for file management of the access target file.

[Claim 2]

The flash memory file storage device described in Claim 1 characterized by the fact that the control means controls the data distribution means so that data of different files do not coexist in the memory area of the minimum deleting unit of the flash memory element.

[Claim 3]

The flash memory file storage device described in the Claim 1 or 2 characterized by the fact that the control means controls the data distribution means so that the second function is selected for data of continuous p pieces of storage capacity units in the file, and that the first function is selected for data of one storage capacity unit

[Claim 4]

The flash memory file storage device described in the Claim 1 or 2 characterized by the fact that the storage capacity unit and the minimum deleting unit of the flash memory element are equal to each other.

[Claim 5]

In an information processing device having as its components a central computing/processing means which processes programs and data, a clock oscillation means which drives the central computing/processing means, a file storage means having flash memory as its storage medium, a file storage control means which controls access of the flash memory of the file storage means, an information processing device loaded with a flash memory file storage device characterized by the fact that the same signal or synchronized signal with clock signal generated by the clock oscillation means is input to the file storage control means, and that the central computing/processing means and the file storage control means perform synchronized actions to send/receive file data.

[Claim 6]

The information processing device described in the Claim 5 characterized by the fact that when the one-time data access width is different between the central computing/processing means and the file storage means, the file storage control means is equipped with a means to output a status indication signal indicating whether file data are being processed or the processing is complete, the central computing/processing means has a function wherein it receives the status indication signal, if it indicates that it is in process, it halts the process until the processing is complete, the file storage control means has a data bit width control means which adjust the number of data bits dealt with by the file storage means to the processing data bit width of the central computing/processing device, the data bit width control means shows to the central computing/processing means that processing is taking place using the status indication signal, and allows it to stand by during the period necessary for adjusting the data bit widths, and performs data exchange by adjusting the processing data width of both.

[Claim 7]

The information processing device described in the Claim 6 characterized by the fact that the file storage means is composed of the number of flash memory groups necessary for generating data of the same number of bits with the data access width of the central computing/processing means, wherein sequential order is regulated for the memory groups, storing files over multiple storage capacity units following this regulated order of memory groups, and when the number of storage capacity units increases by updating a once-stored file, the increase is secured from the memory group regulated in order as the next of memory group where the last data of the file were stored in the previous storing.

[Claim 8]

In an information processing device equipped with a central processing device and a file storage device, characterized by the fact that multiple flash memory elements are used as the storage media of the file storage device, where the minimum deleting unit of data of the flash memory is made to be equal to the storage capacity unit for file management, and if the number of storage capacity units when the central processing device requests file access is plural, the multiple flash memory elements are accessed simultaneously, and if the number of storage capacity units is one, one piece of the multiple flash memory elements is accessed.

[Claim 9]

In a file storage device having flash memory as its storage medium, a flash memory file storage device characterized by the fact that if the storage capacity of a file to record extends over multiple storage capacity units, it is equipped with a chaining information storage means which records information in the physical location where the next storage capacity unit is stored for each storage capacity unit.

[Claim 10]

In a memory element where multiple memory chips are embedded in a package, a memory element where the memory package is equipped with the corresponding number of input/output data terminals to the total number of data inputs/outputs of all the memory chips, equipped with a data control means which controls switching/connecting input/output data of each memory chip and data input/output terminals and a control signal terminal for instructions from a memory user to the data control means, making it possible to switch/connect data of the multiple memory chips upon instructions of the control signal terminal to connect input/output data of an arbitrary memory chip to an arbitrary terminal of the input/output data terminals.

[Claim 11]

In a memory element where multiple memory chips are embedded in a package, a memory element where the memory package is equipped with the corresponding number of input/output data terminals to the total number of data inputs/outputs of all the memory chips, equipped with a data control means which controls switching/connecting input/output data of each memory chip and data input/output terminals and a control command setting means for instructions from a memory user to the data control means, making it possible to switch/connect data of the multiple memory chips upon instructions of the control command setting means to connect input/output data of an arbitrary memory chip to an arbitrary terminal of the input/output data terminals.

[Detailed Explanation of the Invention]

[0001]

[Field of Industrial Application]

The present invention relates to information equipment loaded with a file storage device having flash memory as its storage medium, and especially relates to a file access method of file storage devices suitable for information equipment where high speed file access is demanded.

[0002]

[Prior Art Technology]

In today's general information equipment starting with personal computers, auxiliary storage devices are almost indispensable peripheral equipment. Moreover, in general, auxiliary storage devices are contained in most equipment, and arranged so that the users can always deal with files of large capacities.

[0003]

Recently, there are increasing cases where personal computers of notebook size and palm-top type are used, and portability is coming to be regarded important. Therefore, in place of magnetic disk storage devices which are weak in terms of vibration and have a large power consumption, file storage devices having semiconductors as their storage media are drawing attention. For example, disclosed in Pat Open Hei 2-292798 Public Report is a technology of a semiconductor file storage device using a flash memory as its storage medium.

[0004]

Flash memory is nonvolatile memory which can be electrically rewritten, and because capacity increase and cost reduction are possible, it is one of the most effective memories as storage media of semiconductor file storage devices. The technology in the Pat Open Hei Public Report uses this memory to solve the problems in constructing a file storage device and devices for improving the ease of use. For example, it proposes a method of overcoming the shortcoming of flash memory in which the elements become degraded by frequent rewriting, a method of increasing the speed in a phase called, deleting necessary for rewriting a flash memory, etc. Also, as an interface with information equipment which becomes a host, it proposes it to be the same as that for magnetic disk devices, directed towards the construction of systems that can replace magnetic disk devices.

[0005]

[Problems overcome by the Invention]

The semiconductor file storage device of prior art technology uses the pre-existing interface bus of the host information equipment and places importance on compatibility with magnetic disk devices. Although this makes the semiconductor file storage device to be easily accepted by the users, in return for placing importance on compatibility with magnetic disk devices, no consideration is made on superiority of semiconductor storage elements to magnetic disk devices.

[0006]

Namely, although semiconductor storage elements enable very high speed data access because they are static recording media unlike magnetic disk devices which pick up data from rotating disks and perform writing, there is the problem that the superiority of this high speed access cannot be utilized with the same interface as with magnetic storage devices.

[0007]

It is an objective of the present invention to construct an interface which achieves high-speed access performance to the maximum by making semiconductor storage elements to be its storage media and to provide new auxiliary storage devices which improve the access performance, which is a shortcoming of magnetic disk devices.

[0008]

[Problem Resolution Means]

The flash memory file storage device by this invention is characterized by the fact that in a flash memory file storage device which is a file storage device storing data through an X-bit data bus and uses flash memory elements

whose minimum deleting unit is larger than the X bits and access width is x bits ($x = X/p$; p is an integer equal to or greater than 2), it is equipped with a flash memory device consisting of p sets of flash memory element groups accessible simultaneously, dividing means which divide data on the X -bit data bus into at least p parts, a data distributing means which has at least a first function to make p pieces of x -bit data obtained by the division correspond to one set of the flash memory element groups and a second function to make each of the p pieces of x -bit data correspond to another set of the flash memory element groups, and a control means which controls the data distribution means so that the first and second functions are switched according to the number of storage capacity units for file management of the access target file.

[0009]

[Operation]

In magnetic disk devices used in the current personal computers, data access is slower than the memory access of the personal computer and does not have to operate in synchronization with the CPU in the personal computer. Therefore, the give and take of data of magnetic disk devices are performed on an asynchronous bus. When a semiconductor is made to be a storage medium, because it can follow operations of the CPU, it becomes significant to make it a synchronous operation.

[0010]

However, as a problem in such a case, there is the issue that the CPU processing bus width and the data access bus width of one chip of flash memory are different. Although this bus width difference can be solved by using memory chips in parallel in cases of DRAMs etc., flash memory has a fixed deleting unit where the minimum deleting unit is 512 bytes, for example. In this case, if multiple chips are utilized in parallel, the capacity of a deletion unit that is deleted at once becomes ($512 \times$ the number of parallel chips) bytes. Thus, the deleting target capacity is so large that data other than the deleting target are also deleted.

[0011]

On the other hand, because personal computer file management currently regards this 512 byte unit as one sector, changing this storage capacity unit for file management (also called the file management unit, below) should be avoided. Therefore, in accessing one sector which is the file management unit, if attempting to utilize four flash memory chips in parallel, the minimum deleting unit becomes larger as a result, and rewriting the sector will require deleting four times the capacity of one sector as described earlier. In order to avoid such a situation, if the CPU has a 32-bit bus and the flash memory has an 8-bit bus for example, one chip needs to be accessed four times sequentially to deal with the 32-bit bus of the CPU.

[0012]

However, when continuously accessing four sectors or more, there is no problem even if four chips are accessed simultaneously in parallel for each four-sector data and 32 bits from the CPU data bus are accessed simultaneously with a one-time access. Namely, if 512 bytes in the first sector are written by 32 bits at a time over 16 times, $128 (16 \times 8)$ bytes are dispersed to each step. Next, 512 bytes in the second sector are written by 32 bits at a time over 16 times. In the same way, 512 bytes in the third sector, and further 512 bytes in the fourth sector are written. At this point of time, all the capacity of the minimum deleting units of four chips is filled with data. Therefore, there is no problem in deleting this four-sector capacity at once when rewriting this file.

[0013]

Thus, simultaneous four-step parallel access is repeated for each four sectors of a file. If there are data left which do not fill the last four sectors, other measures will become necessary.

[0014]

If there are three sectors left, because 32 bits of the CPU cannot be divided by 3, they are separated into two sectors and one sector for handling.

[0015]

If there are two sectors left, or the file is two sectors in size in the first place, 32-bit data from the CPU are divided in two, and two chips are simultaneously accessed by 16 bits each at a time. The 512 bytes in the first sector are written onto two chips by 16 bits at a time over 32 times, where 256 (32×8) bytes are dispersed onto each chip. In the same way, the 512 bytes in the second sector are written onto two chips by 16 bits at a time over 32 times.

[0016]

If there is one sector left, or the file is one sector in size in the first place, 32-bit data from the CPU are divided into four, and one chip is accessed by 8 bits at a time. The 512 bytes in the one sector are stored in one sector of the chip over 64 times.

[0017]

As with the above, changing the method of storing data according to the number of continuously accessed sectors contributes to an access speed increase and solves the deletion problem accompanying flash memory. Namely, in either storing method, because it is guaranteed that only data of the same file are stored inside the minimum deletion area of a chip, the problem accompanying the deletion of flash memory is solved.

[0018]

Here, although the flash memory is operated synchronously in this invention, delays occur according to the access scheme. For this problem, a wait status can be arbitrarily requested to the CPU by using a CPU having a ready signal input. CPUs having a ready signal input are popular today. For example, it is installed in all CPUs having 16 bit processing or higher by Intel Corp., which are the most popular as CPUs mounted in general-use personal computers. Because the CPU halts its processing cycle by negating this ready signal, if reading/writing of access data has not been completed, this signal can be just negated, and when the access is complete, the CPU resumes execution of the processing by asserting it. Because this is a control which becomes possible only when the file storage device is in a synchronous operation with the CPU, inputting synchronized clocks to the both at this time to make them operate synchronously is indispensable. By these controls, even when the data width per single processing is different between the CPU and the flash memory, it can be easily dealt with by letting the CPU wait until the data widths are adjusted.

[0019]

It is usually impossible in file management to separate anything dealt with in a continuous sector once. Namely, a file system can only access in file units. Therefore, when accessing multiple sectors continuously, if multiple chips are accessed simultaneously at the time of writing, using the multiple sectors in parallel, the file data as they were written can be obtained at the time of reading through the same parallel access.

[0020]

As in the above, according this invention, in an information processing device mounted with file storage devices having flash memories as its storage media, synchronous control with the system is performed so that the maximum performance can be drawn in its access performance which is highly superior to magnetic disk devices. In such a case, the CPU is put in a wait status upon necessity by a ready signal input to the system CPU to adjust the timing. Even when the system data bus width and the number of access data bits of one chip are different, it becomes possible to access at as high a speed as possible.

[0021]

The effect is especially great in reading/writing of large capacity files whose access speed is easily recognized by the user, namely in continuous sector accesses. Also, whereas the system data bus width and the number of the access data bits of one chip are different according to the purpose, performance, and age, highly-flexible configuration is made possible to deal with various things. Also, it can be applied to methods to access memory with interleaves to increase the speed.

[0022]

According to the configuration equipped with a chaining information, the access procedure from the system is simplified to support a high-speed access. Then, the file management itself can be simplified, and simplification of control circuit, control program, etc. can be achieved.

[0023]

Also, by being equipped with a data distribution function inside a memory element, reduction of peripheral circuits and data processing speed increase can be achieved.

[0024]

[Embodiments]

Embodiments of the present invention are explained hereafter, with reference to the drawings.

[0025]

First of all, shown in Fig. 1 is a standard personal computer configuration where the flash memory file device of this invention is installed. In the figure, 1 is a CPU which in charge of processing data and programs, and its data bus width is assumed to be 32 bits. No. 2 is a clock generator which generates a synchronous clock for the whole system. No. 3 is a system internal bus containing a data bus, an address bus, memory commands, I/O commands, etc. No. 4 is a file control circuit which performs file management and memory control of the flash memory file device of this invention. No. 5 is a flash memory array which becomes a storage medium of the flash memory file device, where the number of access data bits per chip is 8. No. 6 is a main memory control circuit which manages/controls the system main memory, and 7 is DRAM which is the main memory. No. 8 is a peripheral I/O bus control circuit, where a display control circuit 9 and a display 10 are connected as one of peripheral I/O devices. Listed as other peripheral I/O devices are a communication device 11, an external large capacity storage device 12, etc.

[0026]

Usually, the peripheral I/O bus control circuit contains another clock generation circuit, and these peripheral I/O devices operate based on that clock cycle. However, there can be one which is directly connected to the internal system bus 3 for speed enhancement and operates in synchronization with the CPU 1. No. 13 is a clock signal supplied to each circuit containing the CPU 1 for synchronizing circuits connected to the internal system bus 3. Note that it does not need to be entirely the same with the clock supplied to the CPU 1 but can be a divided one depending on the circuit as far as it is synchronized.

[0027]

No. 14 is a ready control signal input to the CPU 1, where the status indication signals output from individual circuits are general-controlled in a ready control circuit 15 and input to the CPU 1. No. 16 is a control circuit of input devices for the user of this system to instruct the desired processing, and 17 is an input device. In the figure, the input device is assumed to be a keyboard, and its control circuit 16 is assumed to be a keyboard controller (KBDC).

[0028]

Next, actions of the system in Fig. 1 are explained. During normal operation, the CPU 1 computes/processes programs and data stored in the main memory 7 to execute the process that the user instructed through the input device 17, and displays the result on the display 10. Also, if necessary, it starts up the communication device 11 and stores a large capacity of data to the large capacity external storage device 12. Then, in retrieving or storing files, it operates the flash memory file device consisting of the file control circuit 4 and the flash memory 5. When booting up the system also, the system program is loaded from here. During these operations, synchronous operations are performed by the clock 13 generated by the clock generator 2, and if there occurs a need for a certain circuit to request a stand-by to the CPU 1, that circuit request a CPU stand-by to the ready control circuit 15, and it is conveyed to the CPU 1 by negating the ready signal 14. The CPU 1 will continue to stand by until the ready signal 14 is asserted again. At this time, the file control circuit 4 can perform a control of increasing/decreasing the CPU stand-by time by the number of files requested to the CPU 1. This mechanism is explained with reference to Fig. 2.

[0029]

Figure 2 is a drawing to explain the internal configuration of the flash memory file device. In the figure, 3, 4, 5, and 13 are the same as those in Fig. 1, and the following are internal components of the file control circuit 4. No. 21 is a register group which performs interfacing with the system bus 3, no. 22 is a status register for reporting the status of the file control circuit 4 to the CPU 1, 23 is a register for setting the starting sector number to access, 24 is a register for setting the ending sector number to access, 25 is a command register which instructs the processing requested by the CPU 1 as a command code, and 26 is a data register which becomes a window of data give and take with the system bus. 27 is a controller which takes a general control of internal controls of the file control circuit 4, which is ideally a programmable, intelligent LSI such as a one-chip microcomputer. No. 28 is a control circuit which performs control of the flash memory 5 which is a storage medium, 29 is a data control circuit which performs control of data to read from or write to the flash memory. No. 30 is a DMA control circuit for performing memory access at high speed, and the system clock signal 13 is input to this circuit. No. 31 is a status indication signal output to the ready control circuit 15, no. 32 is a local bus inside this flash memory file device, and 33 is a control signal and address for accessing the flash memory.

[0030]

Next, actions of the flash memory file device in Fig. 2 are explained. When a necessity to access with the flash memory file device arises, the CPU 1 performs the access via the system bus 3. In order to do so, first the content of the status register 22 is read out to check whether it is in an accessible status or not. Then, the sector to access next is set to the starting sector register 23 and ending sector register 24. Then, it writes the command code (read or write) of the requested access to the command register 25. Then, the status register 22 is read again, and if the access is possible, data are written to or read from the data register 26.

[0031]

At this time, the controller 27 manages these interface registers 21 to respond to requests from the CPU 1. Namely, it reads the starting sector register 23, ending sector register 24, and command register 25, grasps the access content of the flash memory 5, write a code indicating the current status to the status register 22, and reports to the CPU 1.

[0032]

Because it is expected that the controller 27 is too slow in its operation speed to perform directly flash memory access for responding to the access requests from the CPU 1, the configuration is made so that the DMA control circuit 30 performs flash memory access at high speed and performs give and take of data with the system bus 3. The controller 27 will perform setting the access content, starting up the DMA, etc. to the DMA control circuit 30 and the memory control circuit 28 for that purpose.

[0033]

The DMA control circuit 30 performs address generation and timing generation for performing the DMA, and the memory control circuit 28 generates access signals follow that timing. The flash memory 5 performs give and take

of data with the data control circuit by these input signals. The data control circuit 29 performs data generation according to the number of sectors to access.

[0034]

For example, in the case of write access of one sector, data sent by one-time access from the system bus 3 are adjusted to the number of writing bits of one chip of flash memory. In this embodiment, because the data bus width of the system bus 3 is set to 32 bits, 32 bits of data are obtained through a transfer by one access. Then, because the data width of one chip of flash memory is set to 8 bits, the sent data are divided into four to write to the flash memory 5. Therefore, the processing of dividing 32-bit data into four 8-bit data is performed by the data control circuit 29 using a latch circuit. Also, in the case of read access of one sector on the contrary, the flash memory 5 is read accessed four times to prepare 32-bit data for one bus transfer. The wait time of the system bus occurring at this time is generated by a stand-by request to the CPU 1 by a status indication signal 31 generated by the data control circuit 29.

[0035]

On the other hand, in the case of access to multiple sectors, the data control circuit 29 adjusts the data to latch for increasing the access speed and execute it normally. For example, if it is a continuous read access of four sectors, because 32-bit data are completed by simultaneously reading four chips of 8-bit access flash memory 5, the wait time of the system bus 3 can be greatly reduced. Note that four-chip simultaneous write access needs to be performed at the time of writing to the flash memory 5. Otherwise, the order of data will come to differ and the file data will become abnormal. Note that because it is common to perform management with file units in systems dealing with file storage devices, it is usual to read-access the same number of sectors with the number of sectors when they were written, and if the access schemes according to the number of sectors are made entirely the same between writing and reading, there will be no need to record the information. Namely, if it is an access scheme of continuous writing of five sectors, the first four sectors are simultaneously written to four chips in parallel, and the remaining one sector is divided in four to one chip, adopted in both reading and writing for example, and normal file data can always be accessed. Here, to make sure, information on the data storing scheme may be recorded in data in sector units stored in the flash memory 5. As its record location, if there exists a redundant area other than the data storing area of the flash memory, storing there is appropriate, and if there is no redundant area, a separate storing area is installed for recording.

[0036]

A continuous access of six sectors is dealt with a parallel access of four sectors and a parallel access of two sectors.

[0037]

Next, these access signals and data control method are explained in even more detail with reference to Fig. 3.

[0038]

Shown in Fig. 3 is a configuration where the bit width of the system bus is set to 32 bits, and that of the flash memory to 8 bits. In the figure, numbers that already appeared before are identical to those explained so far. Newly, 41 is a counter for generating addresses of the DMA control circuit 30, where the system clock 13 and I/O access (command) signals or memory access (command) signals of the system bus 3 are input and counting up is performed in synchronization with this. No. 42 is a start-up register for DMA control where the local bus 32 of the controller 27 is connected, and a desired DMA transfer can be started by writing a code to this register. No. 43 is a sector register which is connected to the local bus 32, too, and DMA transfer of an arbitrary sector number can be performed by writing the sector number to access. In the actual operation, written value of this sector number is input to the memory control circuit 28 which is used for generating the upper-order address of the flash memory and chip select signal. No. 44 is a timing circuit which generates a timing signal for synchronizing in each control circuit at DMA transfer. No. 45 is a memory address generated by the memory control circuit 28 based on values of the counter 41 and sector register 43. No. 46 is a memory control signal generated by generation of the memory address 45. 51, 52, 53, and 54 are data latches of one byte (8 bits) each, constituting data latches of four bytes (32

bits) for data width conversion between 32-bit data and 8-bit data. When data of the system bus 3 are denoted as D0~D31, the latch 51 latches D0~D7, the latch 52 D8~D15, the latch 53 D16~D23, and the latch 54 D24~D31, respectively. 55 is a latch signal generation circuit for these data latches. 56 is a data control setting register which is connected with the local bus 32 and sets data width and data order. In this embodiment, generation of a latch signal and its timing are instructed to the latch signal generation circuit 55 by setting one of "1", "2", and "4" to this data control setting register 56 as the number of continuous access sectors at read access. Then, latch signals input to data latches 51, 52, 53, and 54 are 57, 58, 59, and 60, respectively. For example, if "1" is set, latch signals 57, 58, 59, and 60 are output sequentially one by one, and data from one chip of flash memory are accessed four times to generate 32-bit data which are output to the system bus 3. Also, if "2" is set, latch signals 57 and 58 are output simultaneously and afterwards latch signals 59 and 60 are output simultaneously, which are performed alternately, and data from two chips of flash memory are accessed twice each to generate 32-bit data which are output to the system bus 3. Also, if "4" is set, latch signals 57, 58, 59, and 60 are output simultaneously, and data from four chips of flash memory are accessed only once to generate 32-bit data which are output to the system bus 3.

[0039]

On the other hand, during a write access, latch signals are always sent to data latches 51, 52, 53, and 54 simultaneously to latch 32-bit data from the system at once. No. 61 is a data distribution circuit which distributes data stored in data latches 51, 52, 53, and 54 or data from the memory 5. No. 62 is a 32-bit data bus connecting the flash memory 5 and the data distribution circuit 61. No. 63 is a read/write signal for determining the data direction of the data distribution circuit 61, which should better receive supply from the command register 25 of the interface register 21. The data distribution circuit 61 is made as a bidirectional buffer, where one side is connected to data latches 51, 52, 53, and 54, and the other side to the flash memory 5. On the flash memory 5 side, the input is in 32 bits, and all the chips of the flash memory 5 are divided into four groups, which are divided into separate bit groups (Bits 0~7), (Bits 8~15), (Bits 16~23), and (Bits 24~31), and input as 32 bits in total. Then, the direction is determined by whether it is read or write, and partitioning of each data are determined by the setting content of the data control setting register 56.

[0040]

Specific explanations are given on this distribution of data using Fig. 12, Fig. 13, and Fig. 14. These figures show data distribution examples of the data distribution circuit 61 for different values set to the data control setting register 56. Figure 12 shows data distribution when the number of continuous access sectors is 1, Fig. 13 when the number of continuous access sectors is 2, and Fig. 14 when the number of continuous access sectors is 4, respectively, and there are four kinds of data distributions in each case depending on which of the four memory groups to access. Also, read and write are distinguished. When the number of continuous access sectors is 1, four system cycles make one access, when the number of continuous access sectors is 2, two system cycles make one access, and when the number of continuous access sectors is 4, one system cycle makes one access.

[0041]

When the number of continuous access sectors is 1, classification into four kinds is made depending on where in the divided memory to access, and when the number of continuous access sectors is 2 or 4, classification into four kinds again is made depending on which memory group is set as the starting point. Namely, the starting point can be adequately determined depending on how the memory is used, and by doing so the bias of used memory groups can be prevented. For example, although the data distribution method can be simplified if the memory group 1 is always set as the starting point, it is expected that the ratio of usage of the memory group 1 becomes high and there occur biases in both the amount of use and the frequency of use. If a bias occurs, one memory group eventually becomes unavailable, making it impossible to perform high-speed write access when the number of continuous sectors is 4. Therefore, it is made possible to set the starting point in any memory group.

[0042]

Note that although in reading when the number of continuous sectors is 1 or 2, data connection lines in the distribution circuit are made separate for each cycle in the figure, because the latch signal is not output to other than the target latch, the connection line may be shared instead of separating for each cycle. Namely, in reading of A

where the number of continuous sectors is 1 (Fig. 12), whereas the distribution is performed in a sequential cycle with latch 1 – memory group 1, latch 2 – memory group 1, latch 3 – memory group 1, and latch 4 – memory group 1, connection lines to all the latches may be connected to 1 of the memory group in all cycles. This is because even if data lines are connected, there is no influence if no latch signal is output.

[0043]

More specifically, referring to Fig. 12 in the write access of one sector for example, one byte connected one of the access target flash memory groups among the 32-bit data bus 62 is written to the flash memory with the data latch 51 at the first time, the data latch 52 at the second time, the data latch 53 at the third time, and the data latch 54 at the fourth time, which is repeated from 51 sequentially. In the read access, one cycle of data are distributed to each latch by the latch signal.

[0044]

In the write access of two sectors, namely, if it is a write with a data width of 2 bytes, from Fig. 13, as 2-byte data connected to the corresponding two memory groups among the 32-bit data bus 62, data are received from the data latches 51 and 52 at the first access, 53 and 54 at the second access, and distributed alternately afterwards. On the other hand, in the read access the order becomes reversed, and data are distributed from the corresponding two memory groups to the data latches 51 and 52 in the first access, 53 and 54 in the second access, and this is repeated.

[0045]

In the write access of four sectors, namely, if it is a write with a data width of 4 bytes, the buffer direction becomes the one from the data latch to the flash memory 5, the 32-bit data bus 62 is connected to the data latches 51, 52, 53, and 54 sequentially from its top memory group, and 32-bit access is completed by one-cycle access. In reading, the direction becomes the opposite.

[0046]

In order to perform the actions, the controller 27 needs to set appropriate values to the registers explained so far before starting up the DMA control circuit 30.

[0047]

When multiple memory groups are accessed simultaneously in order to increase the speed as described earlier, addresses given to the individual memory groups are investigated below referring to Fig. 5.

[0048]

In Fig. 5, 81~84 indicates memory groups 1~4, respectively. The same figure (1) shows a state where a certain amount of file writing has been performed, and the same figure (2) shows a state where a certain file has been updated and its file size has increased. The data code (m-n) shown in the figure indicates a file number m and a sector number n in each file. For example, (3-2) indicates the second sector of the file number 3. Note that this figure shows area (having a capacity of one sector or more) of which memory group is used how as a whole, and in actuality, at a continuous access of two sectors or more, instead of content of only a single sector being stored in a one-sector capacity area of each memory group, contents of multiple sectors are stored dispersed. This point will be described in detail later in the explanation of file management.

[0049]

In Fig. 5 (1), in the access of file number 1 (all sectors coded as 1-n), in a simultaneous access from 1-1 to 1-4, because these are arranged in the same line, same address can be given to these memory groups. However, in file number 2, if 2-1 to 2-4 are used for a simultaneous access, the memory group 4 needs to be given a different address from other memory groups. This means that the same number of address buses with the number of memories need to be installed. Or, there needs to be some way to give different addresses to the memory groups. In order to avoid

this, an access scheme needs to be taken in accesses of file number 2 that 2-1 is accessed alone, 2-2 and 2-3 are simultaneously accessed, and 2-4 is accessed alone again. However, this will generate a waste in increasing the speed. Then, an example of configuration for giving different addresses is explained using Fig. 9.

[0050]

Figure 9 is an example of configuring an address generation circuit inside the memory control circuit 28 when the number of memory groups is set to 4. In the figure, 201~204 are four memory groups constituting the flash memory 5, where 201 is a first memory group a, 202 is a second memory group b, 203 is a third memory group c, and 204 is a fourth memory group d. 205 is a latch circuit b of the high-order address given to the memory group b, 206 is a latch circuit c giving to the memory group c, and 207 is a latch circuit d giving to the memory group d. 208 is the address bus of the memory control circuit 28 (corresponding to 45 in Fig. 3), and 209 is the low-order address of the address bus 208. The number of its address bits is set to the number of address bits which corresponds to that for accessing the storage capacity of the data deletion minimum unit or the minimum unit for file management of the flash memory chip. For example, if the data deletion minimum unit is 512 bytes, it becomes 9 bits. Only when the data deletion minimum unit of the flash memory chip is smaller than the minimum unit for file management, it is effective to make it the number of address bits which corresponds to the minimum unit for file management. 210 is the high-order address of the address bus 208 excluding the low-order address 209. Note that even higher-order address which is unnecessary in accessing the memory groups is removed. No. 211 is a high-order address stored in the address latch b for accessing the memory group b, 212 is a high-order address stored in the address latch c for accessing the memory group c, and 213 is a high-order address stored in the address latch d for accessing the memory group d. No. 214 is a memory control signal for the memory group a201, 215 is a memory control signal for the memory group b202, 216 is a memory control signal for the memory group c203, and 217 is a memory control signal for the memory group d204.

[0051]

In the configuration of Fig. 9, when accessing the memory group a to the memory group d simultaneously, high-order addresses to access these memory groups are written in advance to the address latches 205~207. Then, when performing the access, because the low-order address 210 is common to all the memory groups, the address bus supplies the address for accessing only the memory group a, and memory groups other than the memory group a are accessed through the high-order addresses from the corresponding address latch circuit. Note that access control is entrusted to the memory control signals 214~217, and if access is only to the memory group a and the memory group b for example, it is arranged so that only the memory control signals 214 and 215 become active. Also, in accessing only the memory group d for example, unless the memory control signal 21 is set active, the memory group a may be give any address without a problem. By this means, it becomes possible to give different addresses to individual memory groups, only if the memory groups are dispersed into 4, data of the same file stored in locations at physically different addresses can be accessed simultaneously, enhancing the contribution of speed increase. In the configuration of Fig. 9, the address latch of the memory group a can be omitted. Of course, if it is effective to install an address latch also for the memory group a, it may be done so.

[0052]

According to the above embodiment, the CPU 1 can perform read/write access of a desired sector at high speed by performing simple specifications to relatively a small number of registers. Also, the file controller is made as a one-chip microcomputer, where fine control can be instructed by software, and data transfer can be performed at high speed by loading a DMA transfer control circuit even if the one-chip microcomputer can only perform slower operations than the system. However, if the operation speed of the one-chip microcomputer is high enough to perform high-speed operations for the system, the DMA control circuit is not necessary, and a configuration where all data transfers are performed by the one-chip microcomputer can also be considered.

[0053]

Also, although an explanation has been given in this embodiment assuming the data buses are 32 bits for the system and 8 bits for the flash memory, also for the CPU 1 of 16-bit operation, CPUs of 64-bit operation, and a flash memory of 16-bit I/O, adjustment to other data widths can be easily performed by changing the number of data

latches, configuration of the data control setting register, and control programs of controllers. The concrete example is shown in Fig. 15. Shown in Fig. 15 are concrete numerical values of hardware configuration in a combination of individual bit widths. The horizontal configuration of the figure is the number of data bits of the flash memory, where 4 bit, 8 bit, 16 bit, and 32 bit are listed as examples. The vertical configuration shows the system data width, where 8 bit for simple information equipment, 16 bit and 32 bit which are the mainstream of the present personal computers, and 4 bit and 128 bit for the future personal computers and high-performance computers are listed as examples. Also, the data distribution circuit listed as a hardware configuration is, if explained using the data distribution circuit 61 of the embodiment, shown in the number of bits of one circle \times the number of circles in the figure. Connection of each circle can be shown in the same idea with Fig. 12 ~ Fig. 14. While the number of latches is increase or decrease of the number of latches 51~54 in the embodiment, if the number of memory data bits is larger than the system data width, the latch location will be configured in the data distribution circuit memory side. The number of memory groups indicates the number of divisions of the memory which is divided in four in the embodiment.

[0054]

Note that the number of distribution of the data distribution circuit, the number of latches, and the number of memory groups are lower limit numbers including this embodiment, and if there is some room in the circuit scale, the number of terminals, etc., increasing them will make a more effective system configuration. That is because increasing the number of memory groups gives more room in selecting memory for storing data, which can prevent biases of memory usage and frequency of use. Also, because the number of memories to access in parallel can be increased, speed increase can be further progressed. Especially, when the system data width and the number of memory data bits are equal to each other or the number of memory data bits is larger, although there is no effect of adjusting the data bus widths at all as described in the embodiment, multiple memory chips can be simultaneously accessed and speed increase by the so-called interleave method can be realized. In this case, at the same time with increasing the number of memory groups, the number of memory data distributions in the data distribution circuit and the number of latches need to be increased.

[0055]

Note that although explained in this embodiment is that the flash memory can follow data transfer from the system, to do so it is preferable that a write buffer be installed inside the flash memory. If it is a flash memory which can perform writing at high speed in the future, this kind of write buffer will become unnecessary. If no write buffer is contained in the flash memory, a write buffer can be installed between the flash memory and the data control circuit. Then, at each write access, writing is performed not directly to the flash memory but first to the write buffer, and after data transfer from the system is complete, writing is performed from the write buffer to the flash memory. The configuration diagram of an embodiment of this case is shown in Fig. 4.

[0056]

In Fig. 4, numbers which already appeared before are the same with those explained so far. As new ones, 71 is a data selector which switches connection with the data distribution circuit 61 depending on whether the access is read or write, 72 is a write buffer of 32-bit width which temporarily stores write data, and 73 is a read/write signal for switching the data selector 71, where a part of data of the command register 25 is better to be input. If the access is write, the data distribution circuit 61 is connected with the write buffer, write data are stored in the write buffer, and the controller performs writing to the flash after data transfer from the system. If it is read, the data distribution circuit 61 is connected directly to the flash memory, and the same operation with the read access in Fig. 3 describe above is executed. By making it the configuration, even using a flash memory which does not contain a write buffer and has a slow writing, high-speed accesses become possible, seen from the system.

[0057]

Next, an embodiment of file management is explained with Fig. 5 as an example. Although embodiments of hardware have been explained so far, how to store actually to the flash memory is described below. This operation is basically performed by the system CPU, system program, and the controller of the flash memory files, where the central issue is operation by software. In the explanation using Fig. 5, instead of the content of the software itself,

shown is how unit capacity area (for one sector) of each memory group of the flash memory as the whole is assigned to a sector of a file as a result of the operation. For each sector of a file, unit capacity area is secured in sequential memory groups. For continuous sectors of four or more, one unit capacity area of sequential memory groups 1~4 are assigned to the four sectors (1-1~1-4 in the figure for example). Note that data in each sector are actually stored dispersed in four unit capacity areas. To two sectors (1-5~1-6 in the figure for example), unit capacity areas of two memory groups are assigned. In this case also, data in each sector are stored dispersed in two unit capacity areas. Assigned to the last sector 1-7 of file 1 is a unit capacity area of a single memory group.

[0058]

Here, an even more detailed explanation is given on dispersed storing of two sectors and four sectors using Fig. 10 and Fig. 11. Shown in Fig. 10 is an example of 4-byte simultaneous write, and Fig. 11 is an example of 2-byte simultaneous write. In the same ways as the above, it is assumed that the host system bus is 32 bits, and that the memory access width of one memory group is 8 bits.

[0059]

Shown in Fig. 10 is an example of 4-sector simultaneous write to the memory groups a~d. In the figure, indicated with 211 is data from the system bus, which are sent with 32 bits (4 bytes) as a unit. Indicated with 222~225 are sequential numbers of 8-bit (1-byte) unit data, 222 is the first byte data, 223 is the second byte data, 224 is the third byte data, and 225 is the fourth byte data. Following them, data up to the 2048th byte are continuously sent as those for 4 sectors. Actually, because of the 32-bit bus, from the first byte to the fourth byte are simultaneously received, and the following are the same. Nos. 51~54 are data latches for temporary storage of data shown in Fig. 3.

[0060]

Because of being simultaneously written 4 bytes at a time, after storing 32 bits, namely 4 bytes, simultaneously to the data latches 51~54, each byte is dispersed to the memory groups a~d and is written one byte each. For the next 32-bit data, each byte is written to the following parts of the memory groups a~d. In this way, when 512 bytes corresponding to 1 sector are written, subsequently data for the second sector are written by 4 bytes at a time to the following parts of the memory groups a~d in the same way. Although Fig. 5 is shown for convenience as if each sector is assigned to a specific memory group, as in the above explanation, actually data in each sector are not stored only in a specific memory group but written dispersed to the memory groups a~d. Note that it is true that data of the whole 4 sectors are stored in an area of 4-sector capacity in the memory groups a~d.

[0061]

Shown in Fig. 11 is an example of 2-byte simultaneous write to the memory group a and the memory group b.

[0062]

Because of being a simultaneously written 2 bytes at a time, after storing 32 bits, namely 4 bytes, simultaneously to the data latches 51~54, these are divided in two by 2 bytes each, the first half 2 bytes are first written to the memory group a and the memory group b by 1 byte each, next the latter half 2 bytes are written to the memory group a and the memory group b by 1 byte each. In this way, when 512 bytes corresponding to 1 sector are written, subsequently data for the second sector are written from the first byte to the following parts of the memory groups a and b in the same way. In this case also, the first sector is not stored only to a specific memory group but written dispersed to the memory groups a and b, and in the same way, the second sector is written dispersed to the memory groups a and b following the first sector.

[0063]

Although writing to a single sector is not shown, individual byte data of the data latches 51~54 are simultaneously stored to a certain memory group sequentially in this case. Only in this case, data of one sector are stored to a specific memory group without being dispersed.

[0064]

As can be seen from the above, the important thing is that there is no data mixed from different files in the area of the minimum deletion unit (512 bytes corresponding to one-sector capacity here) of a flash memory element as a result of the above operations. By this guarantee, it becomes possible to achieve high-speed file data storing by writing multiple bytes simultaneously in parallel as effectively utilizing the storage area of memory when rewriting a file without deleting data of other files.

[0065]

As stated above, Fig. 5 (1) shows a state where a certain degree of file writing has been performed, Fig. 5 (2) a state where a certain file has been updated and the file size has increased afterwards. In Fig. 5 (1), storing is performed so that different memories are assigned in the order of file numbers to be stored and the order of sector numbers. In principle, they are packed to leave no space so that there will be no waste in terms of management.

[0066]

As shown in Fig. 5 (1), if file sectors are stored in memory groups by packing sequentially and continuously so that no vacant space occurs, when a file size has increased by updating the file, there occurs cases as shown in Fig. 5 (2) where continuous storing in terms of physical storage location is impossible. Even in such a case, storing is performed so that the order of memory groups becomes continuous. Namely, if the file number 4 is taken as an example, while five sectors from 4-1 to 4-5 are stored in (1), if two sectors are added in (2), because 4-1 is started from the memory group 3, areas in the memory groups 2 and 3 are secured for the increased portion of 4-6 and 4-7. When writing a file whose number of sectors is increased, 4-5 and 4-6 are treated as 2-sector continuous sectors at this time, and 4-7 is treated as a single sector. Namely, added sectors also receives a treatment as continuous sectors together with already-stored sectors. In order to perform smoothly these storing and reading out in file management, it is believed to be more appropriate as a hardware configuration that specification of access sectors from the system should be the starting sector and the number of sectors and grasping the physical storage locations should be performed inside the file system. Namely, whereas specifying access files from the system should be simplified as much as possible for the speed increase, if physical locations are scattered, specifying the physical locations becomes complicated. Therefore, the ending sector 24 shown in Fig. 2 should be removed in this sense. Then, by storing information indicating file chaining in an redundant area of the memory to store information other than data if it exists, or in another storage means if it does not exist, because if the file starting number is specified, all the physical locations of sector numbers following it become clear by chaining, a continuous access becomes possible.

[0067]

Shown in Fig. 6 is an example of chaining information. In the figure, 85 is file data stored in memory, which are data of file number 4 and sector number 5. 85 is the data of file number 4 and sector number 6 that follow them. And 87 is data of file number 4 and sector number 7 that further follow them. Note that although these data are arranged in the order of the sector numbers, they may be mixed in the actual storage state. Namely, data may be stored scrambled by accessing multiple chips simultaneously through a continuous sector access. However, because numbering of data storage by byte is necessary even in such a case, sector numbers are very important. 88 is a physical address indicating the physical location on memory where the file data 85 are stored, where the number "3" on the left side is the memory group number, the number "5" on the right side is the address in that memory group. 89 and 90 are physical addresses of the file data 86 and the file data 87 in the same way. 91 is chaining information indicating the physical address where the next sector of the file data 85 are stored, and because the file data 85 are data of file number 4 and sector number 5, the chaining information 91 indicates the physical address where data of file number 4 and sector number 6 are stored. The number on the left side is the memory group number, and the one on the right side is the address in the memory group. In the same way, chaining information 92 indicates the physical address where data of file number 4 and sector number 7 are stored, and chaining information 93 indicates a case where the next sector does not exist. Namely, it can be told that the file of file number 4 ends with 7 sectors. If this chaining information is attached with file data, the system CPU does not have to grasp the physical location of file storage but can take a scheme of simply accessing the file itself. Then, by the controller of file control performing physical accessing of memory as referring to the chaining information, even with a file whose physical storage locations are not continuous but scattered, a continuous access becomes possible. In that case, a

configuration to input different addresses for simultaneous access of different memory groups is necessary. Note that as stated earlier in Fig. 9, low-order address for the one-sector access can be shared, necessity is only for the high-order address for more than that. According to embodiments using chaining information above, access specification from the system can be simplified to enable the execution with smaller amount of information.

[0068]

Next, an explanation is given on an embodiment of embedding various kinds of functions in a memory element itself.

[0069]

Figure 7 is a configuration diagram of a memory element where multiple memory chips are contained in a package and an additional circuit is installed. In the figure, 101 is a memory package, 102~105 are memory chips having the same function, 106~109 are input/output data terminals of the memory package 101, each of which can perform data input/output of one memory chip. For example, if the memory chip 102 has 8-bit data input/output, 106~109 come to have $8 \text{ bits} \times 4 = 32 \text{ bits}$ of data input/output terminals in total. No. 110 is a signal input terminal which specifies the path to connect these memory chips with input/output terminals, 111 is a specification signal, and 112 is a connection path setting circuit. A user of this memory 101 executes an access after setting 109 with the signal input terminal 110 from the data input/output terminal 106. It is considered as an example that this setting is selected from connection paths such as those in Fig. 12, Fig. 13, and Fig. 14 stated earlier. In the memory 101, set data connection is sent as the specification signal 111 to the connection path setting circuit 112, the connection setting circuit 112 connects the memory chips 102~105 with the data input/output terminals 106~109 according to the setting, and realizes connection of data in the memory chip with the desired data bus.

[0070]

Figure 8 is a configuration example of a memory element which performs setting of data connection from a user by a command input. In the figure, 113 is a command control circuit which creates the specification signal 111 from a register of the command set value and its set value. 114 is a data line where the user set the command set value using a part of the data input/output terminal. Others are the same as those with the same numbers in Fig. 7. A user selects a data connection path with Fig. 12, Fig. 13, and Fig. 14 as an example, sets as a command code from a part of data bus 114 to the command control circuit 113, and sends the specification signal 111 which corresponds to the connection path setting circuit 112 from the command code set in the command control circuit 113. Below is the same as the explanation of Fig. 7. According to this embodiment by Fig. 7 and Fig. 8, another embodiment explained so far can be realized as a memory element, and there is an efficacy that reduction of peripheral circuits can be achieved. Note that although in this embodiment multiple chips and control circuits are to be loaded in a memory package, by uniting these into one chip, miniaturization and speed increase can be promoted.

[0071]

[Efficacy of the Invention]

According to the present invention, it becomes possible in a flash memory device to achieve high-speed file data storing by writing multiple bytes simultaneously in parallel as effectively utilizing the storage area of the memory when rewriting a file without deleting the data of other files.

[Brief Explanation of the Drawings]

[Fig. 1] A block diagram showing a system configuration which realizes this invention.

[Fig. 2] A block diagram showing the configuration of an embodiment of the flash memory file device.

[Fig. 3] A block diagram of the main section for explaining actions of the embodiment.

[Fig. 4] An explanatory drawing showing data control when a slow-writing flash memory chip is used in the embodiment.

[Fig. 5] An explanatory drawing of an execution example of storing files to the memory in the embodiment.

[Fig. 6] An explanatory drawing of an execution example of storing chaining information in another embodiment of this invention.

[Fig. 7] An explanatory drawing of a memory configuration example where the data distribution function by the signal terminal input specification is installed in the memory element.

[Fig. 8] An explanatory drawing of a memory configuration example where the data distribution function by the command setting input specification is installed in the memory element.

[Fig. 9] An explanatory drawing of address connection of the memory groups in the embodiment.

[Fig. 10] An explanatory drawing of actions at 4-byte simultaneous write in the embodiment.

[Fig. 11] An explanatory drawing of actions at 2-byte simultaneous write in the embodiment. (system bus 32 bit.)

[Fig. 12] An explanatory drawing of data distribution at memory 1-chip simultaneous access in the embodiment.

[Fig. 13] An explanatory drawing of data distribution at memory 2-chip simultaneous access in the embodiment.

[Fig. 14] An explanatory drawing of data distribution at memory 4-chip simultaneous access in the embodiment.

[Fig. 15] An explanatory drawing of hardware configurations in various configurations of the system bus and the memory data bus.

[Explanations of the Codes]

1: CPU, 2: Clock generation circuit, 3: System bus, 4: File control circuit, 5: Flash memory, 6: Main memory control circuit, 7: Main memory, 8: External I/O control circuit, 9: Display control circuit, 11: Communication circuit, 12: Large capacity external storage device, 13: System clock, 15: Ready control circuit, 16: KBDC, 21: I/F register, 22: Status register, 23: Starting sector register, 24: Ending sector register, 25: Command register, 26: Data register, 27: Controller, 28: Memory control circuit, 29: Data control circuit, 30: DMA control circuit, 31: Status indication signal, 32: Local bus, 41: Counter, 42: Startup register, 43: Sector register, 44: Timing control circuit, 45: Address, 46: Control, 51: Latch circuit, 52: Latch circuit, 53: Latch circuit, 54: Latch circuit, 55: Latch circuit generation circuit, 56: Data control setting register, 61: Data distribution circuit, 63: Read/write signal, 71: Data switching circuit, 72: Write buffer, 73: R/W signal, 85: Data (4-5), 86: Data (4-6), 87: Data (4-7), 91: Chaining information, 101: Memory package, 110: Specification signal input terminal, 112: Connection path setting circuit, 113: Command control circuit, 201: Memory group a, 202: Memory group b, 203: Memory group c, 204: Memory group d, 205: Address latch b, 206: Address latch c, 207: Address latch d, 208: Address bus, 209: Low-order address, 210: High-order address, 214-217: Memory control signals.

(Continuation of the front page)

- (72) Inventor: Jun Kitahara
c/o Microelectronics Equipment Development Laboratory
Hitachi, Ltd.
292, Yoshida-cho, Totsuka-ku, Yokohama City, Kanagawa Pref.

- (72) Inventor: Yasuhiro Hida
c/o Microelectronics Equipment Development Laboratory
Hitachi, Ltd.
292, Yoshida-cho, Totsuka-ku, Yokohama City, Kanagawa Pref.

- (72) Inventor: Kazunori Furusawa
c/o Semiconductor Design Development Center
Hitachi, Ltd.
20-1, Johsui-Honcho 5-chome, Kodaira City, Tokyo